APPLICATION-INDEPENDENT DOCUMENT STORAGE USING A GENERIC MARKUP LANGUAGE

By

TONY VINCENT HARRISON

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1995

Dedicated to Mom,

Dad, and Ewell.

ACKNOWLEDGEMENTS

I would like to thank Dr. Watson for the opportunity to work with him on this research project. He has helped me in more ways than he knows. This technology is the future in information management. Thanks again, Dennis. I would also like to thank Dr. Mishoe for taking over for Dr. Shoup as my major professor. He has also been a great help to me as a friend and student. Special thanks to Dr. Beck for help in understanding the database and retrieval process with FAIRS, and the overall CD-ROM project. Also, I wish to thank Jeff Nelson, David Williams, Ling Li, and the entire FAIRS staff for their input. Thanks to Dr. Peart for help with learning systems, for the process has been used by me in this project and others. To Dr. Kilmer, I would like to thank him for his patience throughout this long process. I would like to thank Mary Cilley for her work with the FAST-WP development process and editing, Steve Eissinger with WP2SGML, and Michael Harper with retrieval software conversion. I would also like to thank Drs. Shoup and Isaacs, whom I consider as my mentors during my initial years here at the University of Florida. I would also like to say thanks to all the faculty and staff in the Agricultural and Biological Engineering Department. I have so many fond memories of them that are too numerous to

list. Finally, to Cecil and Mary Harrison, and Dalton and Bernice Harrison, thanks for the support.

TABLE OF CONTENTS

Abstract of Dissertation Presented to the Graduate School of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

APPLICATION-INDEPENDENT DOCUMENT STORAGE USING A GENERIC MARKUP LANGUAGE

By

Tony V. Harrison

May 1995

Chairperson: Dr. J. W. Mishoe
Major Department: Agricultural Engineering

Documents are generally stored in a proprietary software format on a specific computer platform. Users who have computer hardware and software different than those that developed the documents must use manual processes such as rekeying the document to access the information. The objective of this research project was to store documents in a format that allows any computer hardware or software to use the information in Florida Cooperative Extension Service (FCES) documents. Fifty documents were randomly selected from FAIRS' DISC8 (a CD-ROM produced by the Florida Agricultural Information Retrieval System (FAIRS) at the University of Florida (UF)) for model development. A document analysis on the FCES publications produced a tree structure that identified document elements and their hierarchical relationships. An International Standard (ISO 8879-1986)

known as Standard Generalized Markup Language (SGML) was used to represent FCES publication structure as a model. The model (Document Type Definition (DTD)) was developed based on the tree structure and the Association of American Publishers (AAP) Article model. The FCES model consisted of elements in the AAP model having the same definition as those in the tree structure and unique elements required by UF. A commercial parser verified that the model conformed to SGML rules and syntax. Each FCES publication was tagged with Florida's Authoring System Tools for WordPerfect (FAST-WP), whose styles were developed from the structural properties in the model. The tagged FCES publications were then converted to SGML instances (tagged text files) based on the structure and content of the model. Each instance was parsed with a commercial parser to verify that the model was an adequate representation of the structure of FCES publications. The FCES model was then found to be application-independent after converting the instances into FAIRS DISC8 and DISC9, Multimedia Viewer, and Guide retrieval system format for on-screen display. The FCES model developed in this research has been incorporated into a vertically integrated electronic information system used by FAIRS to deliver FCES information by CD-ROM and World Wide Web.

CHAPTER I
INTRODUCTION

Justification

A significant problem facing institutions is the inability to retain and share knowledge in a form independent of specific computer systems. The distribution of this vast pool of knowledge in electronic documents is primarily as printed material, although electronic files may be available from a publisher. For decades the distribution of federal documents has been as paper documents and on microfiche (The Office of Technology Assessment, 1988). The cost of distributing information as printed documents is a concern not only of government agencies but also of private businesses. For example, according to Robert Bennett, second vice president at the Travelers Corporation, only 15 percent of the $300 million budget for printing and publishing was for the data-center or printing-center. The remaining 85 percent was associated with the costs of weighing, storage, and people that sort mail in the printing and publishing process (Francis, 1990). Bennett believes electronic publishing can reduce a significant portion of the printing and publishing budget at Travelers Corporation.

The Florida Cooperative Extension Service (FCES) has not been immune to this problem. The high cost of distributing printed documents has led to efforts to distribute information electronically through computer systems. FCES produces about 500 printed publications per year. Budget cuts and increasing publishing costs moved FCES to place all publications on the Florida Agricultural Information Retrieval System (FAIRS) (Beck et al., 1994). FAIRS provides delivery of all FCES documents in electronic form to growers, county extension specialists, researchers, and homeowners. FAIRS uses hypertext, full-text search, and browsing information retrieval strategies to retrieve information from over 700 megabytes of data. The same word processing files used in the printed publication process are delivered through the FAIRS database. However, the conversion of electronic documents into the FAIRS database format was a laborious process. A text editor divided each document into sections based upon structure and content. Besides the labor requirement, any revision of an electronic publication required the repetition of the entire conversion process. These restrictions gave the impetus to automate the entry of structure and content from publications into electronic information systems.

Electronic publishing typically begins with the use of word processing software to develop documents. Word processing software is interactive, self-contained, display-oriented, text-formatting and editing programs used to

generate electronic documents (Rahtz, 1987). However, there is often a limited ability to transfer these documents directly into an electronic information system, because word processing software uses different coding schemes to represent a document's structure (hierarchical organization, depicted by title and headings) and content (i.e., text, figures, tables). However, software systems for retrieval of information typically use a database structure for storage (Beck and Watson, 1992). These two representations of document structure and content are incompatible, producing transferability and portability problems between the word processing files and electronic information systems.

Retaining document structure and information content during changes in computer hardware and software is critical for long-term successful electronic information system development. As computer hardware and software constantly evolve and change, knowledge must outlive the technology used to develop and initially deliver the information. Yuri Rubinsky (1989) says that "Although one cannot anticipate future publishing requirements and technologies, a plan can be developed to recycle information. The best way to do this is to store information in a standardized way, independent of any particular technology or presentation method (page 9)."

Overall Objective

The overall objective of this research project is to model the structure and represent the data of technical publications in an electronic form that is independent of any computer hardware or application.

Specific Objective Number One

The first specific objective is to model the structure of a set of technical publications, and represent the content of the publications in an application-independent form.

Specific Objective Number Two

The second specific objective is to verify the model by automating and testing a process of using FCES publications in SGML form for electronic storage and delivery.

The remainder of this dissertation is organized into five chapters. Chapter II provides an overview of generic markup description, uses, and applications. Also included is a review of several commercial and academic formatting models, and hypertext and hypermedia models used for document conversion. Chapter III describes the procedures for accomplishing the overall and specific objectives in Chapter I. Chapter IV describes the model development process for converting FCES publications into an application-independent format. Chapter V summarizes the results of verifying the model's application-independence by using retrieval software

to present document information. A summary of the authors work, conclusions and recommendations for future work in modeling documents are in Chapter VI.

CHAPTER II
REVIEW OF LITERATURE

Computers as Information Tools

Computer Storage

Computers store information in electronic form, such as the customer database of a business that includes each customer's name, address, and telephone number. Businesses can develop multiple applications such as billing, sales promotions, and customer service when information is organized. Files, chapters, page number, and series of pages are ways to organize both structural and contextual information in electronic documents (Graphic Communications Association, 1991). The content of electronic documents can vary, and include text, images, graphics, spreadsheets, and voice, which can impede transfer of information (Ansen, 1989). However, the key process is the electronic storage and retrieval of the information for human consumption.

Computer Recognition of Document Information and Format

Text is presented in two-dimensional form as information for human consumption (Horak, 1984). For example, technical writers can prepare electronic documents in a structure for

reader interpretation of the information. The structure aids a reader in finding desired information. However, computers only process the information while humans interpret it based on the structure.

When computers are used for document processing, they are primarily used for either printing or viewing a document. For example, one could use information in a printed software manual describing installation procedures for a particular software package. The structure of the manual describes the organization of the information. From this organization, subjects are recognized based on formatting of areas such as titles, where sentences and words end, key words, and punctuation. Coombs et al. (1987) suggest that those documents with both accurate and descriptive markup can be ported from one computer system to another.

Descriptive markup for each subject area in a document allows information to be processed in many ways. However, while documents usually have no explicit structure, database software can retrieve information by particular field names, such as customer name, due to their explicit structure. While the computer only sees the information as an electronic document, humans can distinguish between the information and format of the document. Computers with this ability could treat any electronic document as a database of information. This would allow the computer to find information inside a

manual, handbook, or other document and use it for many different applications.

Desktop publishing software and some word processors enable publishers to set up standard print styles so documents from different authors can be incorporated into a uniform series. These word processors also render the final printed text easier for readers to interpret by making the structure more apparent (Wilson, 1991). For example, a word processing file can allow different computer hardware to use and exchange information representation. First, the electronic file contains codes to specific page size, fonts, and position of text. Proper conversion or duplication of print presentation on a different system requires the same word processing software, printer, and perhaps soft fonts. Second, computers allow the use of many different proprietary software languages and packages. Incompatible computer hardware or software formats require manual intervention by users to use the information. This restricts the electronic access to information that is inside these documents. Finally, accessing the information in electronic documents from different institutions and companies can be very difficult. Valuable time has to be spent explaining the different versions, and possibly helping make the document compatible with another computer system. Upgrading hardware or software systems could be less desirous and cause incompatibilities when there are many documents in an existing format. None of

these concerns are new. Smith (1985) quotes a January 1979 report by the UK National Computer Users' Forum addressing these concerns. The report describes the multiple character sets as the biggest problem when interchanging data. The primary causes of these multiple character sets are the observance of poor standards.

Generic vs. Specific Markup in a Computer Generated Document

The hierarchical organizations of information in electronic documents aid a reader's comprehension of the material. In printed form, a reader can quickly scan a document and understand its structure by viewing the different typefaces and sizes for various levels of information. For example, a large bold font with centered text could represent the highest level headings and a left-justified medium-size font could represent lower level headings. These typographical conventions provide visual cues for the reader, and are important for video display of an electronic document. For example, one could use the highest level headings as a table of contents, with a hyperlink to text for each heading. However, the font and text justification used by one author for first level headings may be the same format used by another author for third level headings. These structural ambiguities prevent modeling a set of documents for use in electronic information systems.

Requiring authors to use the same word processor would allow the production of a model describing the structure and appearance of a set of documents. The model would define specific font and positioning codes (or specific markup) to describe each structural element (i.e., generic markup such as title and author). The explicitness of the model allows software to convert from the file format of the word processing program to a format electronic information systems could understand. However, a model of a set of documents must account for differences in specific formatting codes and preferences before possible use in an electronic information system.

A layer of abstraction can provide a way to account for different formatting codes and preferences in an electronic document. Goldfarb (1980) describes this layer of abstraction as markup. The markup should include first the separation between the model and specific formatting codes, and then the processing functions on structural elements in a document. Generic markup, for example, identifying a first level heading as Head 1 instead of with a large, bold, centered font, provides the needed layer of abstraction. Many word processing programs provide generic markup with a feature commonly called style sheets or styles. Style sheets or styles separate structure and text (content) in a document (Stein, 1991), allowing the formatting or processing of subject areas as needed. Style codes used for printed

documents can serve as a way to introduce generic markup into an electronic document (Cilley and Watson, 1992a and 1992b). The generic markup allows further processing of a document for database storage or other processing such as CD-ROM (Cilley et al., 1990). For example, the authors of an electronic document are identified with an "author" style code instead of specific font information. At the time of printing or display, specific markup replaces the generic codes based upon the style.

A system using generic markup was initiated at the University of Florida for a CD-ROM project (CD-ROM Implementation Group, 1990). The approach of the developers was to add generic codes to word processing documents as an aid to knowledge acquisition. Adding the same generic codes to the same subject areas in multiple documents provides a structure that can be modeled. Development of the document model would allow the conversion of similar documents into files containing both structure and content. This separation gives a set of similar documents an explicit structure that electronic information systems can use, for example, to perform a query search or display document information on-screen. Standard Generalized Markup Language (SGML) (ISO 8879-1986) provides the basis for structural model development.

Standard Markup Languages

History

Publishing companies recognized the need for a standard specifying document architecture as early as the 1960s (Rodgers, 1989). In September of 1967, William Tunnicliffe of the Graphics Communications Association (GCA) suggested using generic coding as descriptive tags to separate information (content) from format (Goldfarb, 1990). In 1969, Charles Goldfarb of IBM developed a Generalized Markup Language (GML)(Goldfarb et al., 1970)(Goldfarb, 1980) to integrate law office information systems (Goldfarb, 1990). Goldfarb's work served as the basis for the international standard, Standard Generalized Markup Language (ISO 8879-1986). Another approach for specifying document architecture is The Office Document Architecture.

ODA/ODIF

The Office Document Architecture and Office Document Interchange Format (ODA/ODIF) is an approach for document interchange currently under development (US Department of Commerce, 1988). Ansen (1989) describes the architecture as a set of standards for both structuring and encoding documents for interchange between dissimilar systems. It is a draft international standard (DIS 8613) under review by the National Institute of Standards and Technology (NIST). Painter (1989)

suggests that ODA looks at document structure both logically (e.g., sections) and as a layout view (i.e., physical view decides how the content appears). Scheller (1988) outlines the following differences between SGML and ODA:

1. ODA restricts attributes to those specified by the standards, while SGML enables the definition of any desired attribute.
2. SGML documents have no semantics defined in the standard, while ODA documents contain semantics for document representation.
3. ODA restricts the content of documents to the standard, while SGML has no restrictions.
4. ODA documents are interpreted by machines and required special input systems, while SGML has no such restrictions.
5. Interchange of ODA documents between different personnel needs no agreement, while SGML documents can only be interpreted within special applications environments.
6. ODA documents are represented by a special formatter or the semantics of the layout description in a document formatting language. The formatter and document class describe the representation of SGML documents.

Scheller (1988) provides the most crucial difference between ODA and SGML as "the fact that ODA facilitates the interchange of documents with restricted functionality between any partner in an open computer network, whereas SGML

documents can only be interchanged within clearly defined applications areas but are not subject to restrictions with respect to functionality (page 142)." He also says that the number of representational possibilities, content types, context-dependent layout descriptions, and automatic generation of both tables of content and references are absent when using ODA in the technical, scientific, and publishing area.

SGML as an International Standard

In the early 1980s, the International Standard Organization (ISO) began preparing standards to allow transfer of multiple document types over varied computer systems (Bryan, 1988). In December of 1986, ISO issued its standard for document representation known as the Standard Generalized Markup Language. The standard committees who deal with the SGML standard areas are as follows:

1) Joint Technical Committee 1 (JTC1) for Information Processing.
2) Sub Committee 18 (SC18) for Text and Office Systems.
3) Working Group 8 (WG8) project 15 for Computer Languages for Processing Text.

The project editor of the standard is Charles Goldfarb of IBM Corporation in the United States.

Other SGML Standards or Reports Being Reviewed within the group that developed ISO 8879-1986 (Smith, 1989a)

1) Document Style and Semantic Specification Language (DSSSL - ISO 10179) - This standard provides a language to describe the translation of SGML markup in a document to a specific format. The simplest application would be a style sheet. This goes in a separate specification than the DTD. It allows the exchange of an SGML file among different systems, and has specific DSSSL representations for the document.
2) Font Information Interchange (ISO/DIS 9541) - Problems occur when exchanging a page or document in DSSSL form from one computer to another when printers are different. For example, a Helvetica 14-point font may look different from one printer to another. Thus, there has to be a standard way of describing the fonts for printed pages of a document to be identical from system to system.
3) Guidelines for SGML Syntax-Directed Systems - This technical report specifies a series of guidelines for the capabilities of an SGML syntax-directed editing system.
4) Retroactive Conversion - This technical report deals with insertion of tags into existing text, whether in databases or word processing files that

do not understand SGML. It looks at SGML features that can reduce the markup needed within a document.

5) SGML Document Interchange Format (SDIF - ISO 9069) - This ISO standard allows the interchange of an SGML document by means of open systems interconnection (OSI) techniques.

6) Standard Page Description Language (SPDL - ISO 10180) - XeroX and Adobe are developing this as a standard postscript language. It would provide a way to exchange finished pages of an electronic document between computer systems in a standard way. For example, an application receives a document page from one computer to another for identical printing of the page.

7) Techniques for using SGML (ISO/DTR 9573) - These techniques describe the design of document type definitions, including mathematics. Criticisms outlined by Smith (1989a), which have been directed to this area, include not taking advantage of database publishing and mathematics.

There are several ongoing projects for SGML conformance (Graphic Communications Association, 1991). First, there is an initiative from the executive committee in the Graphics Communication Association (GCA) to create a laboratory worldwide to test SGML software for conformance to

international standards. Second, there is a project for the development of the binary encoding of SGML (SGML-B). It entails a one-to-one translation between an SGML file and SGML-B file to enable quick access time by a computer. This is important for CD-ROM production. Currently, sequential coding requires building indexes for fast access to information. SGML-B will allow CD-ROM production personnel to place a binary file directly on a CD.

Second, a Hypermedia/Time-Based Subset (HyTime) solution for hypertext was first published by the ISO late in 1992 (ISO/IEC 10744:1992). HyTime was added to provide ways for different information to coexist and work together in an everchanging environment. The combination of HyTime and SGML provides greater information management among different media such as textual information, audio, animation, and digital information. As of this writing (August, 1994), there are no fully HyTime-conforming applications in the marketplace.

What SGML Is

SGML is a standard for full-text database publishing (Smith, 1986b) that defines the character set for processing information safely over any system (SoftQuad, 1991). It provides a descriptive language for modeling document architecture in specific syntax. In SGML terminology, a structural model of a set of documents is a document type

definition (DTD). Documents converted to an SGML format based on the structural model are instances.

For model development, SGML is the standard for defining the element names (i.e., subject areas), and their order, location, frequency, and relationships within a document. An SGML model explicitly follows the structure of a set of documents. For example, an SGML model could require the generic name "chapter" to be placed at each chapter heading in a set of documents.

Modeling the structure of a set of documents allows conversion of documents into a standard file format interpreted by most electronic information systems. ASCII, the American Standard Code for Information Interchange, is a standard character set (i.e., file format) that SGML can use to represent DTDs (models) and instances. For example, software could convert word processing files with generic markup into ASCII format (instances) based on the model (DTD). Electronic information systems then translate the instances into their respective format for video display.

What SGML Does Not Mean

First, SGML is not a tag set or programming language. It does not require specific elements in a model or provide a set of rules to mark up a document. Second, SGML does not define the meaning of structural elements within the document. Rather, SGML provides the document structure required by

electronic information systems to access the information within a document. Third, an SGML file will not describe how to process an element. An element is open to any processing application. The SGML application, a program that uses the tagged SGML file, decides how each element will be processed. The data or content of an element is not important, for there is no way to verify that the information between the tags is appropriate for that type of element. For example, an entire document can be placed under the element "title" in the SGML file (instance).

How Does SGML Describe Structure?

SGML defines document structure formally so a computer application can use the information. However, the structure must not be too restrictive, but flexible enough to represent several types of documents. An example of being too restrictive might be to define the structure of a set of documents that has 100 chapters, and another structure for documents with 30 chapters. The introduction of multiple chapters into the structure allows both document classes to be defined in the same model.

An SGML model (DTD) normally precedes each tagged text file (instance). This enables computer software to learn the structural properties of the document that follows. An example might be a structural model representing all automobile technical manuals of a particular corporation. The

document type definition (DTD) is the model that defines the structure of the automobile technical manuals. The DTD would appear before any instance representing a tagged automobile technical manual.

How Does SGML Work?

An SGML model provides the names (i.e., generic identifiers), location, order, and frequency of elements in a document. When needed, attributes provide a greater description of elements in the document structure. For example, the sex of element "author" could be either male or female. The element "author" may have an attribute named "sex," which can have attribute values of either "male" or "female."

Thus an element may contain data and/or other structural properties. Each element can contain information (data) and/or be contained within other elements (content model). For example, a chapter may contain other elements such as paragraphs, titles, and sections. There is no limit or restriction as to what type of information can go inside an element. An example may be of an element named videotape. The data within the element may then be a VHS coded tape.

Conceptually, the model is always the same. An SGML processor will read an SGML tagged text file (instance) and perform operations based on the generic markup or coding. The two pieces needed are a brain (parser) and what may be called

a representer (Graphic Communications Association, 1991). The parser reads and understands the instance, then passes the information to the representer. The representer can then, for example, convert the information (instance) into a format suitable for on-screen display, or provide an image to the information for presentation in a publishing system.

An SGML instance is a database of information that does not do anything by itself. The parser reads and understands information in the document by following the SGML model (DTD). The parser reads the DTD to distinguish between information and generic markup (i.e., element names) in the instance. The parser uses the DTD to recognize each element (generic name) in the instance. The instance is considered validated when the parser verifies that each element belongs in the location, order, or frequency found in that instance. An electronic information system that can read SGML instances of a specific SGML model may then use the file as directed. However, a representer can also be used by storage and retrieval systems without the above capability to prepare the information in an instance for their respective use.

A representer prepares information in the instance for use by software such as electronic information systems. Thus, the representer creates a particular representation for the elements. There are many ways to develop a representer, which can be very software intensive. The application that will finally use the information after it has gone through the

representer needs instructions that it can understand. The representer will need a list of elements, their meaning (e.g., <t> - title), their needs, and the specific instructions to give to an application when it receives the elements. The representer expects the proper input, because the parser will verify that the SGML file corresponds to the DTD. Examples of a representer include a converter for formatting codes, database loader, and query search.

An SGML Application

The development of an SGML document model often requires that goals for the SGML application be set and a working group selected. A working group is a select group of individuals involved in the development of an SGML application (model). Given a subset of a class of documents such as fact sheets (Figure A-1, Appendix A), the working group breaks down the documents into pieces (Figure A-2, Appendix A) and develops a tree structure representation (Figure A-3, Appendix A) of the document model. The model (DTD) (Figure A-4, Appendix A), a vocabulary representation of the document structure, is written upon completion of the document analysis and validated for conformance to ISO 8879-1986 standards and SGML syntax. Tagged documents (Figure A-5, Appendix A), known as instances, are validated to ensure conformance (i.e., no errors) with the model (DTD). A set of validated instances provides the explicit structure required by electronic information systems.

SGML applications can be either narrow or broad in scope. The Electronic Manuscript Project of the Association of American Publishers (AAP) (1987) and Computer-aided Acquisition and Logistic Support (CALS) for the Department of Defense are two broad SGML applications. AAP developed SGML applications for book, journal, and article creations between 1983 and 1987. The AAP SGML application standard enjoys wide support in publishing industries such as CD-ROM, and has been adopted as an ANSI application standard (Z39.59). Cover (1992) and <TAG> (SGML Associates, Inc., 1992) list seven document models (DTDs) that are available on the Internet. These include the Text Encoding Initiative (TEI) DTDs, MAJOUR (Modular Application for Journals) DTDs based upon the AAP Article DTD, a HyTime DTD, public DTDs available from Exeter, the CALS-BBS forum, DTDs supporting the AAP/EPSIG manuscript standard, and the "Information Architecture" working group DTD of the OSF Documentation Special Interest Group.

Benefits of Converting Documents Into SGML Instances

Time savings is a major benefit of using generic markup because the document does not have to be coded twice (Graphic Communications Association, 1991). The publication process is also reduced because no further rekeying or proofreading of text is required while initiating the use of a standard approach to the preparation of electronic documents (Smith, 1986a). Smith (1986a) provides an example of personnel

initially using word processing macros to apply specific markup to document information. However, this was reduced to recognizing, then tagging the structural elements with both generic and specific markup. This made the job a lot easier, faster, and requires less specialized labor, resulting in significant cost savings.

SGML also provides benefits for the information retrieval process. Previously, documents were stored in either whole text form such as printed material or as electronic files with "specific markup." SGML aides the information retrieval process through query efficiency and automatic hyperlinking. Query efficiency is improved by selecting specific headings, sections, or topics instead of an entire list of occurrences of a particular subject. The hierarchical structure of SGML documents allows hyperlinks to link together parts of documents such as words, titles and sections.

Hypertext Markup Language

The Hypertext Markup Language (HTML) is based on SGML, and is used to describe the general structure of publications (Lemay, 1995). The structural components of a publication in ASCII format are labeled using tags defined by HTML. Web browsers such as Mosaic (Pfaffenberger, 1994), which is supported by the National Center for Supercomputer Applications at the University of Illinois, provide the network functions to retrieve the HTML documents over the

internet and World Wide Web (WWW). The browser then reads the HTML information and formats the text and images on the screen. The World Wide Web initiative began in 1990, and is a cooperative organization based at CERN, the European Particle Physics Laboratory in Switzerland (Lemay, 1995). However, the tag selection in HTML is very limited. Currently HTML Level One handles headings, paragraphs, images and a few lists. Two other levels of HTML have been proposed. HTML Level Two is similar to HTML Level One, but has additional features to support interactive forms that can provide different options based on a readers' input. HTML Level Three, often called HTML+, will include elements for centered and right-aligned text, tables, mathematical equations, and the alignment of text and images next to each other.

Commercial Document Processing Models

Document preparation and editing involve defining the structure and content of documents, while formatting is concerned with the actual physical layout of a document for both hardcopy and softcopy (Furuta et al., 1982). Formatting documents using generic and specific markup can serve the dual purpose of printed and on-screen display. Commercial products vary widely in applications, from text preparation to conversion of documents to SGML instances. Table 2-1 provides brief descriptions of several commercial products.

Academic and Academic/Commercial Document Processing Models

Academic and commercial products resulting from academic research have been developed for various stages in document processing. A review of several of these models is as follows.

Abstract Document Model

Kimura (1986) presents this document processing system as an interactive document editor based on an expressive document model for paper and electronic documents. Earlier papers have presented concepts of this abstract document model in more detail (Shaw, 1980)(Furuta et al., 1982)(Kimura and Shaw, 1984)(Kimura, 1984). The basis for the document processing system is the notions of abstract and concrete objects, the hierarchical composition of both ordered and unordered objects, component sharing, and reference links (Kimura, 1984). Kimura (1984) also classifies objects in either textual, tabular, mathematical, or pictorial classes. Written in C, a prototype of the system has been in operation since the fall of 1983. The system consists of three major software modules. The graphical abstract document editor (ADE) integrates the abstract object module (AOM) and window object module (WOM), producing the prototype system. Abstract object classes were also developed to write and view technical documents. The uniqueness of the system is the model, interrelationships between windows, unique views generated,

and allowance of structural editing within the document using specific commands (Kimura, 1986).

Andra Text Editor

Andra (Gutknecht and Winiger, 1984) is a modern text editor and formatter for the personal computer Lilith (Wirth, 1981). Professor N. Wirth developed Lilith at the Institute for Informatik of the ETH Zurich from 1977 to 1980. Andra consists of three major parts: input manager, document manager, and display manager. The input manager interprets user input, then translates commands to procedure calls. The document manager maintains the representation of documents. The display manager continuously shows part of the documents being edited on the screen.

Ohio State's Chameleon Project

The Chameleon translation software architecture (Mamrak et al., 1988a, 1988b, 1988c, and 1988d) (Nicholas and Mamrak, 1988) was renamed Integrated Chameleon Architecture (ICA) (Mamrak et al., November 1990). The project studied the different ways that data can be represented (e.g., different word processors) and translated to a desired coding scheme (e.g., SGML format). ICA addresses the broad variety of data representations by the construction and use of data translators (Mamrak et al., May 1987) (Mamrak et al., September 1989). Both the design and implementation of ICA

toolsets occurred over a five-year period (Barnes, July 1990) (Kaelbling, 1987) (Mamrak et al., September 1989) (Nicholas, 1988) (O'Connell, 1990) (Share, 1988a and 1988b). The goal of the developers was to design and implement a code-generating, user-friendly, data-translation architecture that would handle translations for data representations from a selected subset of data objects.

COBATEF System

The COBATEF system is a context-based text formatting system (Peels et al., 1985) consisting of both hardware and software areas of implementation. An automatic text-element recognition mechanism takes advantage of the implicit structure of text, opening the way for a fully-automatic text-processing system. The COBATEF system can recognize text elements by their context in two ways. The document can be scanned for markup for element recognition or by a processing procedure that derives document structure from the content. COBATEF's software package converts the document into its logical structure. It has a horizontal formatter text identification and vertical formatter that produces device-independent print files. Several papers are available that give the hardware developments on the project (Janssen, et al., 1985) (Nijland and Peels, 1985) (Peels, 1984).

FOAM

The FOAM text formatting system (Ganzinger and Willmertinger, 1985), stands for Formatting and Meta-formatting. FOAM was developed specifically to run on available microcomputer systems. FOAM supports meta (description) and text levels of formatting. Descriptions of text and document classes (at meta level) are input to a macro processor, which draws specific formatting styles from a database of macro definitions and generates a specific formatter instance. At the text level, the resultant formatter accepts textual input of the described document class and produces a formatted document based on the formatting styles created at the meta level.

FORMEX

FORMEX (Guittet, 1985) (i.e., the formalized exchange of electronic publishing) was developed to confront problems with recovering varied formats of electronic data and text within the European Community's Office for Official Publications (OP). The Project Management Department in the OP developed FORMEX as a way to store publications in a computer-readable format for information interchange between multiple authors, printers, and computer systems.

FORMEX unified two ways of interchanging electronic information. The first approach was adapted from the common communication format (CCF) developed by UNESCO (1984) and

based on the Format for Bibliographic Information Interchange on Magnetic Tapes (ISO 2709). Output software extracts data from files stored in a mainframe database. A formatter then produces and places the information into a file according to the specifications of the CCF. The resultant CCF file is then validated with a CCF parser. Automatic and manual editing, and pertinent SGML information, are added to enable the file to be upgraded to an FORMEX file. However, FORMEX is not ideally suitable for the transfer of electronic documents consisting of textual information.

The second approach met ISO 8879-1986 standards for text preparation and interchange. It allowed the creation and interpretation of electronic document by humans and computers. Typically a document is produced by an author on a word processor. The document conforms to SGML standards by presenting the information in an explicit format. A formatter converts the document into SGML format based on ISO 8879-1986 standards and the DTD. An SGML parser validates the SGML file and the document information is structured as specified by CCF for upgrading to an FORMEX file.

Integrated System for Complex Computer-Based Documents

Feiner et al. (1981)(1982) developed a system of different programs which drew pictures, composed pages, and graphically specified and presented the contents of pages of computer-based documents. The documents are known as directed

graphs, whose contents are made up of nodes (pages). These pages can be nested in chapters to format documents such as books. The directed-graph structure was developed from previous research on the Hypertext Editing System (Carmody et al., 1969) and FRESS (van Dam and Rice, 1971), also known as File Retrieval and Editing System. These text processing systems have information structuring and retrieval capabilities and are useful for document preparation (Strandberg et al., 1976). The system is a series of programs for modifying and presentation of a document. The processes followed for electronic documents include picture layout, document layout, and document presentation.

Text Editor Lara

Lara (Gutknecht, 1985) is a text editor developed for the Lilith workstation (Wirth, 1981). It succeeds the Andra system (Gutknecht and Winiger, 1984) and does not depend upon a style file. Rather than applying style elements to document structural areas, Lara copies attributes from one place on the computer display to another in the same and other documents. This allows a particular group of documents to achieve the same format. Consistency in the format of displayed text allows a connection with the internal data structure. Thus the internal data structure is not dependent on the editing process, but is inferred from characteristics of the currently displayed text.

Maestro

Maestro, also known as Management Environment for Structured Text Retrieval and Organization, is a model consisting of tools to take advantage of structural knowledge in bibliographic data (Macleod, 1990). Maestro was developed using both conceptual modeling and an object oriented philosophy. It consists of a definitional facility and query language for handling queries and updates. The definitional facility analyzes and constructs a second document that contains a structural representation of the original document. XGML (Exoterica Inc., 1987) was the commercial compiler used in this process. A document developed with the definitional facility consists of content, attributes, and structure of the text. The query language was developed from previous work by the author (Macleod and Reuber, 1987) on document-retrieval systems. The main objective of this process was to develop a language that could naturally handle all text processing applications.

Mixed mode document processing system

Yamada et al. (1987) present this system as an extended document processing model for constructing mixed mode documents. The system contains both structuring and layout editing processes. The structuring process entails scanning a printed document and automatically creating a structured document. A structuring algorithm is used to hierarchically

separate regions such as characters, figures, and tables. The layout editing process consists of both content and structure editors. An interactive pattern recognition process was also proposed. The method consists of both machine-dependent and human-dependent processes that reduce the psychological load on a human operator.

PEN

PEN (Allen et al., 1981), a hierarchical document editor, is a computerized manuscript preparation system for documents containing significant mathematical notation. The interactive formatter provides visual feedback as the author is typing the document. PEN's unique contribution is that it provides notation to simplify mathematical text entry.

TEXTNET

TEXTNET (Trigg and Weiser, 1986) is currently a local level network-based approach for structuring text. The research studied different text organization strategies and their effects on the scientific community. However, it has seen use as an aid for text manipulation. TEXTNET integrates into one approach a local network of both chunks of text in a document and linked documents consisting of on-line literature of scientific nature. The text is stored in a way to make its underlying structure explicit. This provides a way for meaning to be extracted from relationships between chunks of

text. For example, one chunk can support the results obtained in another chunk, whether in the same or different documents.

Other Document Preparation Systems

Furuta (1989) examined the various capabilities, features, and structure of other document preparation systems such as TEX (Knuth, 1984), LATEX (Lamport, 1985), troff (Kernighan, 1981), Scribe (Reid, 1980) (Unilogic, Ltd., 1984), Interleaf (Ilson, 1988), MacWrite (Apple Computer Inc., 1984), XEROX's Tioga (Teitelman, 1984 and 1985), MIT's Etude (Hammer et al., 1981) (Ilson, 1980), and IBM's Janus (Chamberlin et al., 1981) (Chamberlin et al., 1982).

Lee and Malone (1988) explored solutions for computer-based office system problems such as user communication with different templates and document interchange between different word processing programs. The solutions were based on the development of extensions to the Information Lens System (Malone et al., 1987) (Malone et al., May 1987).

Appendix H provides other literature on software development and SGML reviewed but not included in this research. Other software products (not reviewed) using a parser include products from DocuPro, Compugraphic, Frame, Scribe (Smith, 1989b), and Xyvision.

Hypertext and Hypermedia Systems

Bush (1945) is generally credited with the proposing the initial principles on which current hypertext systems are based. Hypertext systems allow one to reference (link) specific segments or the entire current document in question with other on-line documents in a variety of sequences (Newcomb et al., 1991). A true hypertext system should make users feel that they can move freely through the information solely based on their own needs (Nielson, 1990). Hypermedia systems create multi-media (e.g., text, graphics, sound, and executable programs) documents with hyperlinks (Newcomb et al., 1991).

Several first generation (pre 1980's) hypermedia systems include NLS/Augment (Englebart and English, 1968) (Englebart et al., 1973) (Englebart, 1984), FRESS (Meyrowitz, 1986), Thumb (Price, 1982), and ZOG (McCracken and Akscyn, 1984) (Robertson et al., 1981). Conklin (1987) provides a review of these systems, which were primarily mainframe-based systems. Another system from the late 1960's was HES (Carmody et al., 1969) (van Dam, 1988).

The second generation hypertext/hypermedia systems were research oriented systems developed for use on workstations. These included KMS (Akscyn et al., 1988), a newer version of ZOG, Neptune (Delisle and Schwartz, May 1986) (Delisle and Schwartz, December 1986), Intermedia (Garrett et al., 1986)

(Meyrowitz, 1986), and NoteCards (Halasz, et al., 1987) (Halasz, 1988) (Trigg, 1988).

The next generation hypertext/hypermedia systems are currently being developed for use on personal computers. Some of these include Guide (Brown, 1987) (Owl International, 1986), Hyperties (Shneiderman, 1987a and 1987b), and HyperWriter! (Ntergaid Inc., 1992).

Another investigation of hypertext structure and content produced Trellis (Stotts and Furuta, 1988 and 1989), which allowed the author to specify browsing semantics along with structure and content as parts of a document. Delisle and Schwartz (1987), Zellweger (1988), and Trigg (1988) provide other work as a basis for allowing the author/reader to specify traversal paths for hypertext documents.

Products have been developed with hypertext capabilities based on SGML. Smith (1989b) outlines three such products:

1) Idex, a product of Office Workstations Limited, uses hypertext in a multi-user environment.
2) Optical disks as an output medium have been addressed by the Optical Publishing Association.
3) The Silversmith product of Taunton Engineering combines both hypertext capabilities and optical disk production.

Some hypertext/hypermedia systems include those used by Washington University's Manual of Medical Therapeutics (Frisse, 1988) and The Oxford English Dictionary (Raymond and

Tompa, 1988); a system for page-oriented databases (Tompa, 1989); and a system for management of software life-cycle documents (Garg and Scacchi, 1990).

Table 2-1 -- Commercial Document Processing Products.

Company Name	Product Name	Design/ Compile/ Validate Models	Authoring/ Editing/ Documents	Translation/Conversion To SGML				Automatic or Interactive Tagging/ Parsing/ Validation	Create/ Parse/ Validate an Instance	Converts Word Processing Files to SGML Format	Converts SGML Files to Different Formats
				Style Code Based	Logical Text Structure Based	Rule Based	Translation Programming Language and Filters				
Agfa (1991)	Computer Automated Publishing System 6.0x	✔	✔				✔	✔	✔	✔	
Arbortext, Inc. (1994)	Adept Publisher		✔	✔							
	Document Architect	✔									
	Adept Editor		✔					✔			
	Adept PowerPaste					✔		✔	✔		
auto-graphics, inc. (1994)	SGML Smart Editor										✔
Avalanche (1994)	FastTag			✔			✔			✔	
	SGML Hammer										✔
	Interleaf Developer's Kit			✔			✔			✔	
	SGML Developer's Kit			✔			✔			✔	✔
	SureSTYLE	✔		✔							
Data Conversion Laboratory (1994)	MindReader									✔	
	SGMLword										✔
Datalogics (1991)	WriterStation		✔			✔		✔			
	ParseStation								✔	✔	
Day & Zimmermann Info Solutions (1994)	SGML Import Program & Interactive Presentation Manager										✔
Electronic Book Technologies, Inc. (1994)	DynaText		✔								✔

Table 2-1 -- continued.

Company Name	Product Name	Design/ Compile/ Validate Models	Authoring/ Editing/ Documents	Translation/Conversion To SGML				Automatic or Interactive Tagging/ Parsing/ Validation	Create/ Parse/ Validate an Instance	Converts Word Processing Files to SGML Format	Converts SGML Files to Different Formats
				Style Code Based	Logical Text Structure Based	Rule Based	Translation Programming Language and Filters				
Electronic Book Technologies, Inc. (1994)	DynaTag			✔						✔	
Exoterica (1987)	XGML CheckMark	✔						✔			
	XGML Tester	✔									
	XGML Translator	✔					✔	✔			
	XGML Normalizer	✔									
	XGML Validator	✔									
Exoterica (1994)	Omnimark	✔		✔			✔			✔	✔
Folio Corporation (1994)	Folio VIEWS SGML Toolkit	✔									✔
Grif S.A. (1994)	Grif SGML Editor		✔								
InContext Systems (1994)	InContext 2	✔	✔					✔			
InfoAccess, Inc. (1994)	Guide Professional Publisher		✔				✔				✔
Information Dimensions (1994)	BASIS SGMLserver										✔
Interleaf, Inc. (1994)	Interleaf5 <SGML> and Toolkit	✔	✔					✔	✔	✔	✔
Lexicon Systems, Inc. (1994)	ESSE	✔	✔					✔	✔		
Microsoft Corporation (1994)	SGML Author For Word		✔	✔						✔	✔
Microstar Software, Ltd. (1994)	NEAR & FAR	✔									
NICE Technologies (1994)	SGML TagWizard		✔					✔		✔	

Table 2-1 -- continued.

Company Name	Product Name	Design/ Compile/ Validate Models	Authoring/ Editing/ Documents	Translation/Conversion To SGML				Automatic or Interactive Tagging/ Parsing/ Validation	Create/ Parse/ Validate an Instance	Converts Word Processing Files to SGML Format	Converts SGML Files to Different Formats
				Style Code Based	Logical Text Structure Based	Rule Based	Translation Programming Language and Filters				
Officesmiths, Inc. (1991)	Officesmith Markup Language		✔							✔	✔
Passage Systems, Inc. (1994)	PassagePRO									✔	✔
Sema Group (1991)	Mark-It/Write-It	✔	✔					✔	✔		
Shaffstall Corporation (1994)	SGML Translator				✔		✔				
SoftQuad Inc. (1991)	Author/Editor	✔	✔	✔		✔		✔		✔	
Texcel Software GmbH (1994)	Information Manager										✔
US Lynx (1991)	Context-Wise									✔	
WordPerfect Corporation (1993b)	Intelitag		✔	✔		✔		✔		✔	
XSoft (1993)	InContext	✔	✔		✔				✔		
Zandar Corporation (1994)	TagWrite 4 ALCHEMY										✔

CHAPTER III
PROCEDURES

Procedures For Specific Objective Number One

The first objective was to model the structure of a set of FCES publications, and represent the content of the publications in an application-independent form.

The first procedure for specific objective number one was to do a document analysis on a sample set of FCES publications. Fifty FCES extension fact sheets and circulars made up the sample set of publications. The document analysis identified the structural properties of the FCES publications, including the identification, naming, hierarchical order, location, frequency, and interrelationships between elements.

The second procedure for specific objective number one was to develop a model (DTD) that provided a vocabulary representation of the structural elements identified during document analysis. International Standard ISO 8879-1986 (Standard Generalized Markup Language (SGML)) was used to define how structural elements were placed in the model. The DTD structure was compared with other industry standard DTDs for compatibility. Currently, the Association of American Publisher's DTDs (i.e., book, article, and serial DTDs) are the only recognized American standards. Industry standard

DTDs were considered alternatives to in-house development of models when structure was similar. Transferability and portability of information in FCES publications were important considerations during the model development and selection process. XGML Validator (Exoterica Corporation, 1987), a commercial software product, was used to ensure the DTD conformed to ISO 8879-1986 standards and SGML syntax. This was the initial validation step.

Procedures For Specific Objective Number Two

The second specific objective was to verify the model by automating and testing a process of using FCES publications in SGML form for electronic storage and delivery.

The first procedure for specific objective number two was to generate electronic files of the sample FCES publications based on the structure of the FCES model. Currently, WordPerfect (WordPerfect Corporation, 1993a) is the FCES standard for word processing software. FCES authors are using WordPerfect (WordPerfect Corporation, 1993a) as a word processor with an additional pop-up menu (Appendix F) to apply both generic and specific styles as markup. The pop-up menu and associated software tools (macros, styles, and soft fonts) are known as FAST-WP, Florida's Authoring Tools for WordPerfect. After development of the DTD in specific objective number one, FAST-WP underwent extensive modifications to reflect the structural elements in the FCES

model. FAST-WP was then used to tag the structural elements in the sample set of FCES publications.

The second procedure for specific objective two was to convert the tagged FCES publications into SGML format (ASCII tagged text files). In-house software (WP2SGML) created an SGML instance (ASCII tagged text file) for each FCES publication based on the model developed in specific objective number one. XGML Validator Software (Exoterica Corporation, 1987) parsed each FCES instance to verify they conformed to the FCES model.

The third procedure for specific objective number two was to convert the fifty FCES instances into a format that retrieval systems such as FAIRS, Guide (InfoAccess, Inc., 1994), and Mulitmedia Viewer (Microsoft Corporation, 1994) can understand. The conversion of instances to the format of several retrieval systems allowed a subjective evaluation of the model. Each structural element in the model received a ranking for importance in on-screen display. Elements were ranked for on-screen display as extremely important, somewhat important, or not needed for display. The conversion process also determined whether the translation of elements into each retrieval system format would be automatic or manual. The generation of electronic databases from FCES publications in SGML format (instances) was used to verify that the DTD was application-independent and a useful model of FCES publications.

CHAPTER IV
MODEL DEVELOPMENT PROCESS

Currently, no other college institution has a model or model development process that FCES can review for developing application-independent document storage for publications. Appendix B describes an industry process that uses a generalized markup language as the preparatory step for application-independent document storage of publications. An SGML tutorial (Graphic Communications Association, 1991) provided this process of model development and gave some direction for FCES development of SGML applications.

Determine Sample Set of FCES Publications

The number of FCES publications chosen as a representative sample set was fifty. The documents were selected from volumes one and two on FAIRS DISC8. Only documents created after June 1, 1993 were selected to ensure that most publications were tagged with FAST-WP. A total of eight hundred and ninety-five publications fell into this category. Each filename begins with two letters that describe the heading under which the publication can be found. The five digit number following the two letters is the actual document number. A mathematical function @RAND in the Quattro

Pro spreadsheet program (Borland International, Inc., 1992) was chosen to randomly select the fifty documents. Appendix C provides the filenames and titles that were randomly selected.

Results of Document Analysis on Selected FCES Publications

A graduate level course was the initial setting for structural analysis of FCES publications (Harrison et al., 1992). Members of the class consisted of mostly computer and communication specialists. The class was divided into two working groups, with each group having an identical subset of FCES publications. Both groups did a document analysis on the FCES publications and wrote a tree diagram to represent the structure. An attempt was made to develop one tree diagram from the structures of both trees. Generally, each group preferred their own description of the FCES publication structure. Keeping both tree structures would result in the development of two models to represent the same FCES publications. Describing FCES publication structure in two formats would prevent one group from interpreting the others' information. The authors' experience with the class showed that structural specifications for specific informational areas are open to varying interpretations. Usually, the interpretation of structure in FCES publications was dependent upon the person's level of experience, training, personal preference, and knowledge of the publishing environment.

At this point the author did a more extensive document analysis on the set of FCES publications. Structural areas in each document were broken down to the textual (lowest) level, and definitions were given for each area of information. Each definition was based on a review of identical areas of information in the set of FCES publications. Further refinements included the relating of text back to its parent structural elements, merging of similar structural areas, deleting of redundant structural areas, and restructuring and simplification of structure. The primary organization of the first level of structure was a front matter and body matter. These two main elements allow all other informational elements to attach at some level in the structure. The tree diagram and structural definitions were the basis for the FCES model development.

Model (DTD) Development and Selection

It is important at this phase to decide whether one should use a custom model for private use or modify an existing model in the marketplace. Modifying an existing model (DTD) in the marketplace could increase access by the public to information in FCES publications. The information in FCES publications can be accessed and used by those systems that can read and interpret the existing model. Removal of these modifications allows the information to reflect the original model if needed. A review of literature found that

the Association of American Publishers (1987) models are the only models recognized as an American National Standards Institute (ANSI) standard. The AAP models are entitled Book, Article, and Serial. A comparative analysis of the AAP models with the FCES tree diagram and structural definitions were initiated. Upon review, the AAP Article DTD most closely resembled the structural properties and definitions of information in the FCES publications. Appendix D lists the tree diagrams that represent FCES publication structure using the AAP Article model and some unique elements. Figure 1 shows the current FCES model.

The first set of elements unique to the FCES publication model were hyperlink (hyp), link word (lword) and action (act). The AAP Article model does not provide a way to reference a hyperlink in publications. For example, tagging "Figure 1" in the text of an FCES publication as the hyperlink link word (lword) that the retrieval software can do some action (act) on. Two attributes were defined for the element act to describe the type of action to be performed on each hyperlink. The first attribute was named "type." The attribute values of "type" could be text, a graphic, an executable program, audio, a table or a data record. The second attribute for element act was named "descr." This attribute stores in an FCES instance the information that was placed in a comment box for each hyperlink. The descr attribute has a value of character date (CDATA). Hyperlink

elements were placed as an inclusion within the content model of the article element. This allows a hyperlink to occur anywhere within an FCES publication.

The second set of elements unique to the FCES publication model were the rectangular coordinates for simple and complex tables (tdim). The elements provided a way to describe the row height (rowhgt) and column width (colwid) in inches for each row/column combination in simple and complex tables. Their main purpose was to simplify computation of table dimensions. The row height and column width were listed as comma delimited numbers in 1200ths of an inch, and described all columns and rows in a table.

Five attributes were added to complex table header (cth), simple table cells (c) and complex table cells (cte) to aid retrieval software in displaying tables on-screen. The first attribute (shaded) described if the element was shaded or not (y/n). The remaining attributes were named for the top line (topline), bottom line (botline), left line (lftline) and right line (rgtline) that surrounds each of the three elements. The four attributes were given the same attribute value name entitled "name." The name defined whether each line surrounding the three elements was (n)one, (s)ingle, (d)ouble, d(a)shed, d(o)tted, (t)hick or (e)xtra thick. Appendix E defines the elements, attributes and entities in the FCES model.

```
<!DOCTYPE   article
[
<!ELEMENT   article          --   (fm, bdy)       +(fig|fn|hyp)>
<!ELEMENT   fm               --   (tig, au*, pubfm?, abs*)>
<!ELEMENT   tig              --   (atl)>
<!ELEMENT   atl              --   (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*>
<!ELEMENT   (it|b|e1|e2|e3)  --   (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*>
<!ELEMENT   fgr              --   (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*>
<!ELEMENT   f                --   (sup|inf)>
<!ELEMENT   (sup|inf)        --   (#PCDATA)>
<!ELEMENT   au               --   (snm)>
<!ELEMENT   (onm|snm)        --   (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*>
<!ELEMENT   pubfm            --   ((crt|avl)|(aid|issn))*>
<!ELEMENT   (aid|issn)       --   (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*>
<!ELEMENT   avl              --   (onm)>
<!ELEMENT   crt              --   (crd)>
<!ELEMENT   crd              --   (mo?, day?, yr)>
<!ELEMENT   abs              --   (h?, p, (p|(tbl|ctbl)|(l1|l2|l3)|f)*)>
<!ELEMENT   (mo|day|yr)      --   (#PCDATA)>
<!ELEMENT   h                --   (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*>
<!ELEMENT   bdy              --   (sec)+>
<!ELEMENT   sec              --   (st,(p|(tbl|ctbl)|(l1|l2|l3)|f|top1)*, ss1*)>
<!ELEMENT   ss1              --   (st,(p|(tbl|ctbl)|(l1|l2|l3)|f|top1)*, ss2*)>
<!ELEMENT   ss2              --   (st,(p|(tbl|ctbl)|(l1|l2|l3)|f|top1)*, ss3*)>
<!ELEMENT   ss3              --   (st,(p|(tbl|ctbl)|(l1|l2|l3)|f|top1)*)>
<!ELEMENT   st               --   (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*>
<!ELEMENT   p                --   (#PCDATA|((it|b|e1|e2|e3)|fgr)|(tbl ctbl)|(l1|l2|l3)|f)*>
<!ELEMENT   top1             --   (h?,p,(p|(tbl|ctbl)| (l1|l2|l3)|f)*)>
<!ELEMENT   (l1|l2|l3)       --   (lh?, li*)>
<!ELEMENT   lh               --   (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*>
<!ELEMENT   li               --   (p,(p|(tbl|ctbl)  (l1|l2|l3)|f)*)>
<!ELEMENT   fig              --   EMPTY>
<!ELEMENT   fn               --   (p,(p|(tbl|ctbl)  (l1|l2|l3)|f)*)-(fig|fn)>
<!ELEMENT   hyp              --   (lword, act) -(fig  fn|hyp)>
<!ELEMENT   lword            --   (#PCDATA)>
<!ELEMENT   act              --   (#PCDATA)>
<!ATTLIST   act              type   (text|bitmap|exeprgm|audio|table|datarec)#REQUIRED
                             descr  CDATA                                   #REQUIRED>
<!ELEMENT   tbl              --   (no?,tt,tdim?,tby)-(fig|fn|tbl|ctbl)>
<!ELEMENT   no               --   (#PCDATA)>
<!ELEMENT   tt               --   (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*>
<!ELEMENT   tby              --   (th*, tsh*, row*)>
<!ELEMENT   row              --   (tsb?, c*)>
<!ELEMENT   (th|tsh)         --   (#PCDATA|((it|b|e1|e2  e3)|f|fgr))*>
<!ELEMENT   (tsb|c)          --   (p,(p|(l1|l2  l3)|f)*)>
<!ATTLIST   c                     shaded   (y|n)    #REQUIRED
                                  topline  NAME     #REQUIRED
                                  botline  NAME     #REQUIRED
                                  lftline  NAME     #REQUIRED
                                  rgtline  NAME     #REQUIRED>
<!-- The NAME defines whether there is (n)o line, (s)ingle, (d)ouble, -->
<!-- d(a)shed, d(o)tted, (t)hick, or (e)xtra-thick line at the top, left, -->
<!-- right, and bottom of each cell -->
<!ELEMENT   ctbl             --   (cthd,ctby,cthf) -(fig|fn|tbl|ctbl)>
```

Figure 4-1 -- The FCES Model.

```
<!ELEMENT   cthd              --   (no?,ctt?,tdim?,cthr*)>
<!ELEMENT   ctt               --   ((it|b|e1|e2|e3) |f|#PCDATA)*>
<!ELEMENT   tdim              --   (colwid, rowhgt)>
<!ELEMENT   (colwid|rowhgt)   --   (#PCDATA)>
<!ELEMENT   cthr              --   (cth*)>
<!ELEMENT   (cte|ctc)         --   (p|(l1|l2|l3)|f)*>
<!ELEMENT   (cth|ctsb1)       --   (p|(l1|l2|l3)|f)*>
<!ATTLIST   cth                    align    (l|c|r|d)    #IMPLIED
                                   valign   (t|m|b)      #IMPLIED
                                   cb       NUMBER       #IMPLIED
                                   ce       NUMBER       #IMPLIED
                                   rb       NUMBER       #IMPLIED
                                   re       NUMBER       #IMPLIED
                                   shaded   (y|n)        #REQUIRED
                                   topline  NAME         #REQUIRED
                                   botline  NAME         #REQUIRED
                                   lftline  NAME         #REQUIRED
                                   rgtline  NAME         #REQUIRED>
<!-- The NAME defines whether there is (n)o line, (s)ingle, (d)ouble, -->
<!-- d(a)shed, d(o)tted, (t)hick, or (e)xtra-thick line at the top, left, -->
<!-- right, and bottom of each cell -->
<!ATTLIST   ctsb1                  align    (l|c|r|d)    #IMPLIED
                                   valign   (t|m|b)      #IMPLIED>
<!ATTLIST   cte                    align    (l|c|r|d)    #IMPLIED
                                   valign   (t|m|b)      #IMPLIED
                                   cb       NUMBER       #IMPLIED
                                   ce       NUMBER       #IMPLIED
                                   rb       NUMBER       #IMPLIED
                                   re       NUMBER       #IMPLIED
                                   shaded   (y|n)        #REQUIRED
                                   topline  NAME         #REQUIRED
                                   botline  NAME         #REQUIRED
                                   lftline  NAME         #REQUIRED
                                   rgtline  NAME         #REQUIRED>
<!-- The NAME defines whether there is (n)o line, (s)ingle, (d)ouble, -->
<!-- d(a)shed, d(o)tted, (t)hick, or (e)xtra-thick line at the top, left, -->
<!-- right, and bottom of each cell -->
<!ELEMENT   ctby              --   (ctr)*>
<!ELEMENT   ctr               --   (ctsb1?,cte*)>
<!ELEMENT   ctbf              --   (ctc)>
<!ENTITY    amp                          "&" >
<!ENTITY    lt                           "<" >
<!ENTITY    rsqb                         "]" >
<!ENTITY    aacute      SDATA            "[aacute]">
<!ENTITY    Aacute      SDATA            "[Aacute]">
<!ENTITY    auml        SDATA            "[auml]">
<!ENTITY    Auml        SDATA            "[Auml]">
<!ENTITY    eacute      SDATA            "[eacute]">
<!ENTITY    Eacute      SDATA            "[Eacute]">
<!ENTITY    iacute      SDATA            "[iacute]">
<!ENTITY    Iacute      SDATA            "[Iacute]">
<!ENTITY    oacute      SDATA            "[oacute]">
<!ENTITY    Oacute      SDATA            "[Oacute]">
<!ENTITY    ouml        SDATA            "[ouml]">
```

Figure 4-1 -- continued.

```
<!ENTITY    Ouml      SDATA    "[Ouml]">
<!ENTITY    uacute    SDATA    "[uacute]">
<!ENTITY    Uacute    SDATA    "[Uacute]">
<!ENTITY    uuml      SDATA    "[uuml]">
<!ENTITY    Uuml      SDATA    "[Uuml]">
<!ENTITY    bull      SDATA    "[bull]">
<!ENTITY    squf      SDATA    "[squf]">
<!ENTITY    diams     SDATA    "[diams]">
<!ENTITY    cent      SDATA    "[cent]">
<!ENTITY    check     SDATA    "[check]">
<!ENTITY    copy      SDATA    "[copy]">
<!ENTITY    ndash     SDATA    "[ndash]">
<!ENTITY    mdash     SDATA    "[mdash]">
<!ENTITY    deg       SDATA    "[deg]">
<!ENTITY    prime     SDATA    "[prime]">
<!ENTITY    Prime     SDATA    "[Prime]">
<!ENTITY    female    SDATA    "[female]">
<!ENTITY    agr       SDATA    "[agr]">
<!ENTITY    bgr       SDATA    "[bgr]">
<!ENTITY    dgr       SDATA    "[dgr]">
<!ENTITY    Dgr       SDATA    "[Dgr]">
<!ENTITY    mgr       SDATA    "[mgr]">
<!ENTITY    sgr       SDATA    "[sgr]">
<!ENTITY    Sgr       SDATA    "[Sgr]">
<!ENTITY    male      SDATA    "[male]">
<!ENTITY    minus     SDATA    "[minus]">
<!ENTITY    times     SDATA    "[times]">
<!ENTITY    divide    SDATA    "[divide]">
<!ENTITY    plusmn    SDATA    "[plusmn]">
<!ENTITY    le        SDATA    "[le]">
<!ENTITY    ge        SDATA    "[ge]">
<!ENTITY    ne        SDATA    "[ne]">
<!ENTITY    frac12    SDATA    "[frac12]">
<!ENTITY    frac13    SDATA    "[frac13]">
<!ENTITY    frac14    SDATA    "[frac14]">
<!ENTITY    frac23    SDATA    "[frac23]">
<!ENTITY    frac34    SDATA    "[frac34]">
<!ENTITY    Ntilde    SDATA    "[Ntilde]">
<!ENTITY    ntilde    SDATA    "[ntilde]">
<!ENTITY    ldquo     SDATA    "[ldquo]">
<!ENTITY    rdquo     SDATA    "[rdquo]">
<!ENTITY    reg       SDATA    "[reg]">
<!ENTITY    trade     SDATA    "[trade]">
<!ENTITY    iquest    SDATA    "[iquest]">
<!ENTITY    iexcl     SDATA    "[iexcl]">
<!ENTITY    laquo     SDATA    "[laquo]">
<!ENTITY    raquo     SDATA    "[raquo]">
]>
```

Figure 4-1 -- continued.

The AAP Article model provides opportunities to represent multiple elements in the FCES publication model. One such opportunity is the provision in the AAP Article model for up to five different types of lists. The FCES model requires three different lists. The FCES model consists of bulleted (l1), unmarked (l2) and enumerated(l3) lists. The AAP model also provides for three types of emphasis (i.e., e1, e2 and e3) other than elements such as bold (b) and italics (it). Currently the FCES model includes e1 (not currently used), e2 (scientific name) and e3 (reference title) as emphasis types.

The superscript and subscript elements were not initially found in the main part of the AAP Article model, but are required in the FCES model. The two elements are in the AAP math model, which is a part of the AAP Article model. A geometric formula (f) consists of superscript and subscript elements in the AAP Math model. Adding the geometric formula element to the FCES model in the structural locations specified for the AAP Math model allowed the addition of superscript and subscript. Table 4-1 provides a line-by-line explanation of the FCES model, including the new additions.

Table 4-1 -- FCES Model Explanation.

<table>
<tr><th>Line in Model</th><th>Explanation of Content Model</th></tr>
<tr><td><!DOCTYPE article</td><td>A generic name (article) is given for each FCES publication. This is the AAP model name placed at the beginning and ending of each FCES instance.</td></tr>
<tr><td><!ELEMENT article - - (fm,bdy) +(fig|fn|hyp)></td><td>Each FCES publication (article) is composed of front matter (fm), followed by the body (bdy). Both elements are required. Figures (fig), footnotes (fn), and hyperlinks (hyp) can appear anywhere in the front matter or body.</td></tr>
<tr><td><!ELEMENT fm - - (tig,au*,pubfm?,abs*)></td><td>The content model for front matter (fm) is composed of one model group. The order that elements must appear in the front matter (fm) is title group (tig), followed by the author (au) components, publisher's front matter group, and abstract. The title group is required in the front matter, while the author components and abstract are optional and repeatable. The publisher's front matter group (pubfm) is optional and can only occur once.</td></tr>
<tr><td><!ELEMENT tig - - (atl)></td><td>The content model for title group (tig) is composed of one model group. The title group (tig) consists the article title (atl), which is required.</td></tr>
<tr><td><!ELEMENT atl - - (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*></td><td>The content model for article title (atl) is composed of three model groups. The article title (atl) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a geometric formula (f).
(4) a figure reference (fgr).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT it - - (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*></td><td>The content model for italics (it) is composed of three model groups. The italics (it) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a geometric formula (f).
(4) a figure reference (fgr).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT b - - (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*></td><td>The content model for bold (b) is composed of three model groups. The bold (b) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a geometric formula (f).
(4) a figure reference (fgr).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT e1 - - (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*></td><td>Not currently used in model. Bold and italics take care of the straight emphasis in the publications.</td></tr>
<tr><td><!ELEMENT e2 - - (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*></td><td>The content model for scientific name (e2) is composed of three model groups. The scientific name (e2) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a geometric formula (f).
(4) a figure reference (fgr).
The entire content model is optional and repeatable.</td></tr>
</table>

Table 4-1 -- continued.

<table>
<tr><th>Line in Model</th><th>Explanation of Content Model</th></tr>
<tr><td><!ELEMENT e3 - - (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*></td><td>The content model for reference title (e3) is composed of three model groups. The reference title (e3) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a geometric formula (f).
(4) a figure reference (fgr).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT fgr - - (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*></td><td>The content model for figure reference (fgr) is composed of three model groups. The figure reference (fgr) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a geometric formula (f).
(4) a figure reference (fgr).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT f - - (sup|inf)></td><td>The content model for geometric formula (f) is composed of one model group. The geometric formula (f) can consist of either superscript (sup) or subscript (inf).</td></tr>
<tr><td><!ELEMENT sup - - (#PCDATA)></td><td>The superscript (sup) content model consists of text (#PCDATA).</td></tr>
<tr><td><!ELEMENT inf - - (#PCDATA)></td><td>The subscript (inf) content model consists of text (#PCDATA).</td></tr>
<tr><td><!ELEMENT au - - (snm)></td><td>The content model for author (au) is composed of one model group. The author (au) consists of a surname (snm), which is required.</td></tr>
<tr><td><!ELEMENT onm - - (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*></td><td>The content model for name of the organization (onm) is composed of three model groups. The name of the organization (onm) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a geometric formula (f).
(4) a figure reference (fgr).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT snm - - (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*></td><td>The content model for surname (snm) is composed of three model groups. The surname (snm) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a geometric formula (f).
(4) a figure reference (fgr).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT pubfm - - ((crt|avl)|(aid|isso))*></td><td>The content model for publisher's front matter (pubfm) is composed of three model groups. The publisher's front matter group (pubfm) can consist of only one of these structural situations:
(1) copyright notice (crt) or distributor/available From (avl).
(2) article identifier (aid) or international standard serial number (isso).
The entire content model is optional and repeatable.</td></tr>
</table>

Table 4-1 -- continued.

<table>
<tr><th>Line in Model</th><th>Explanation of Content Model</th></tr>
<tr><td><!ELEMENT aid - - (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*></td><td>The content model for article identifier (aid) is composed of three model groups. The article identifier (aid) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a geometric formula (f).
(4) a figure reference (fgr).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT isso - - (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*></td><td>The content model for international standard serial number (isso) is composed of three model groups. The international standard serial number (isso) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a geometric formula (f).
(4) a figure reference (fgr).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT avl - - (onm)></td><td>The content model for distributor/available from (avl) is composed of one model group. The distributor/available from (avl) consists of name of organization (onm), which is required.</td></tr>
<tr><td><!ELEMENT crt - - (crd)></td><td>The content model for copyright notice (crt) is composed of one model group. The copyright notice (crt) consists of copyright notice (date) (crd), which is required.</td></tr>
<tr><td><!ELEMENT crd - - (mo?,day?,yr)></td><td>The content model for copyright notice date (crd) is composed of one model group. The structural elements for the copyright notice date (crd) must appear in the following order: month (mo), day (day), and year (yr). Both month and day are optional and can only appear once. The year is required and is currently the only element used for the copyright notice date.</td></tr>
<tr><td><!ELEMENT abs - - (h?,p,(p|(tbl|ctbl)|(l1|l2|l3)|f)*)></td><td>The content model for abstract (abs) is composed of four model groups. The abstract (abs) can begin with a heading (h). It is optional and can occur only once. Immediately following the optional heading is a paragraph (p) that is required. There are four structural possibilities that can occur following the required paragraph (p):
(1) another paragraph (p).
(2) a simple (tbl) or complex (ctbl) table.
(3) only one of the following lists: bulleted (l1), unmarked (l2), or enumerated (l3).
(4) a geometric formula (f).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT mo - - (#PCDATA)></td><td>The month (mo) content model consists of text (#PCDATA).</td></tr>
<tr><td><!ELEMENT day - - (#PCDATA)></td><td>The day (day) content model consists of text (#PCDATA).</td></tr>
<tr><td><!ELEMENT yr - - (#PCDATA)></td><td>The year (yr) content model consists of text (#PCDATA).</td></tr>
</table>

Table 4-1 -- continued.

<table>
<tr><th>Line in Model</th><th>Explanation of Content Model</th></tr>
<tr><td><!ELEMENT h - - (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*></td><td>The content model for heading (h) is composed of three model groups. The topic paragraph heading (h) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a geometric formula (f).
(4) a figure reference (fgr).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT bdy - - (sec)+></td><td>The content model for body (bdy) is composed of one model group. The body (bdy) can consist of one or more sections (sec).</td></tr>
<tr><td><!ELEMENT sec - - (st,(p|(tbl|ctbl)|(l1|l2|l3)|f|topl)*,ss1*)></td><td>The content model for section (sec) is composed of four model groups. The section (sec) must start with a section title (st). There are 5 structural possibilities that can follow the required section title (st):
(1) another paragraph (p).
(2) a simple (tbl) or complex (ctbl) table.
(3) only one of the following lists: bulleted (l1), unmarked (l2), or enumerated (l3).
(4) a geometric formula (f).
(5) a topic paragraph (topl).
Numbers 1 through 5 are optional and repeatable.
A subsection level 1 (ss1) can follow the five structural possibilities above. It is optional and repeatable.</td></tr>
<tr><td><!ELEMENT ss1 - - (st,(p|(tbl|ctbl)|(l1|l2|l3)|f|topl)*,ss2*)></td><td>The content model for subsection level 1 (ss1) is composed of four model groups. The subsection level 1 (ss1) must start with a section title (st). There are 5 structural possibilities that can follow the required section title (st):
(1) another paragraph (p).
(2) a simple (tbl) or complex (ctbl) table.
(3) only one of the following lists: bulleted (l1), unmarked (l2), or enumerated (l3).
(4) a geometric formula (f).
(5) a topic paragraph (topl).
Numbers 1 through 5 are optional and repeatable.
A subsection level 2 (ss2) can follow the five structural possibilities above. It is optional and repeatable.</td></tr>
<tr><td><!ELEMENT ss2 - - (st,(p|(tbl|ctbl)|(l1|l2|l3)|f|topl)*,ss3*)></td><td>The content model for subsection level 2 (ss2) is composed of four model groups. The subsection level 2 (ss2) must start with a section title (st). There are 5 structural possibilities that can follow the required section title (st):
(1) another paragraph (p).
(2) a simple (tbl) or complex (ctbl) table.
(3) only one of the following lists: bulleted (l1), unmarked (l2), or enumerated (l3).
(4) a geometric formula (f).
(5) a topic paragraph (topl).
Numbers 1 through 5 are optional and repeatable.
A subsection level 3 (ss3) can follow the five structural possibilities above. It is optional and repeatable.</td></tr>
</table>

Table 4-1 -- continued.

<table>
<tr><th>Line in Model</th><th>Explanation of Content Model</th></tr>
<tr><td><!ELEMENT ss3 - - (st,(p|(tbl|ctbl)|(l1|l2|l3)|f|topl)*)></td><td>The content model for subsection level 3 (ss3) is composed of four model groups. The subsection level 3 (ss3) must start with a section title (st). There are 5 structural possibilities that can follow the required section title (st):
(1) another paragraph (p).
(2) a simple (tbl) or complex (ctbl) table.
(3) only one of the following lists: bulleted (l1), unmarked (l2), or enumerated (l3).
(4) a geometric formula (f).
(5) a topic paragraph (topl).
Numbers 1 through 5 are optional and repeatable.</td></tr>
<tr><td><!ELEMENT st - - (#PCDATA|((it|b|el|e2|e3)|f|fgr))*></td><td>The content model for section title (st) is composed of three model groups. The section title (st) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a geometric formula (f).
(4) a figure reference (fgr).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT p - - (#PCDATA|((it|b|el|e2|e3)|fgr)|(tbl|ctbl)|(l1|l2|l3)|f)*></td><td>The content model for paragraph (p) is composed of five model groups. The paragraph (p) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a figure reference (fgr).
(4) a simple (tbl) or complex (ctbl) table.
(5) only one of the following lists: bulleted (l1), unmarked (l2), or enumerated (l3).
(6) a geometric formula (f).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT topl - - (h?,p,(p|(tbl|ctbl)|(l1|l2|l3)|f)*)></td><td>The content model for topic paragraph (topl) is composed of four model groups. The topic paragraph (topl) can begin with a heading (h). It is optional and can occur only once. Immediately following the optional heading is a paragraph (p) that is required. There are four structural possibilities that can occur following the required paragraph (p):
(1) another paragraph (p).
(2) a simple (tbl) or complex (ctbl) table.
(3) only one of the following lists: bulleted (l1), unmarked (l2), or enumerated (l3).
(4) a geometric formula (f).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT l1 - - (lh?,li*)></td><td>The content model for bulleted lists (l1) is composed of one model group. The content model for a bulleted list (l1) begins with a list head (lh) that is optional and can only occur once. The list items (li) must follow the list head (lh), and are optional and repeatable.</td></tr>
<tr><td><!ELEMENT l2 - - (lh?,li*)></td><td>The content model for unmarked lists (l2) is composed of one model group. The content model for a unmarked list (l2) begins with a list head (lh) that is optional and can only occur once. The list items (li) must follow the list head (lh), and are optional and repeatable.</td></tr>
<tr><td><!ELEMENT l3 - - (lh?,li*)></td><td>The content model for enumerated lists (l3) is composed of one model group. The content model for a enumerated list (l3) begins with a list head (lh) that is optional and can only occur once. The list items (li) must follow the list head (lh), and are optional and repeatable.</td></tr>
</table>

Table 4-1 -- continued.

<table>
<tr><th>Line in Model</th><th>Explanation of Content Model</th></tr>
<tr><td><!ELEMENT lh - - (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*></td><td>The content model for list head (lh) is composed of three model groups. The list head (lh) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a geometric formula (f).
(4) a figure reference (fgr).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT li - - (p,(p|(tbl|ctbl)|(l1|l2|l3)|f)*)></td><td>The content model for list item (li) is composed of four model groups. The list item (li) must begin with a paragraph (p). There are four structural possibilities that can occur following the required paragraph (p):
(1) another paragraph (p).
(2) a simple (tbl) or complex (ctbl) table.
(3) only one of the following lists: bulleted (l1), unmarked (l2), or enumerated (l3).
(4) a geometric formula (f).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT fig - - EMPTY></td><td>Not currently used in conversion process.</td></tr>
<tr><td><!ELEMENT fn - - (p,(p|(tbl|ctbl)|(l1|l2|l3)|f)*) -(fig|fn)></td><td>The content model for footnote (fn) is composed of four model groups. The footnote (fn) must begin with a paragraph (p). There are four structural possibilities that can occur following the required paragraph (p):
(1) another paragraph (p).
(2) a simple (tbl) or complex (ctbl) table.
(3) only one of the following lists: bulleted (l1), unmarked (l2), or enumerated (l3).
(4) a geometric formula (f).
The entire content model is optional and repeatable.
There can be no figures (fig) or footnotes (fn) inside a footnote.</td></tr>
<tr><td><!ELEMENT hyp - - (lword,act) -(fig|fn|hyp)></td><td>The content model for hyperlink (hyp) is composed of one model group. A hyperlink (hyp) consists of a link word (lword) and action (act). Both are required and must appear in order. There can be no figures (fig), footnotes (fn), or other hyperlinks (hyp) inside a hyperlink.</td></tr>
<tr><td><!ELEMENT lword - - (#PCDATA)></td><td>The content model for link word (lword) consists of text (#PCDATA). The link word is the word(s) in the publication that references a hyperlink to some type of action (act).</td></tr>
<tr><td><!ELEMENT act - - (#PCDATA)></td><td>The content model for action (act) consists of text (#PCDATA). The action on the link word could be to text, table, executable program, etc.</td></tr>
<tr><td><!ATTLIST act type (text|bitmap|exeprgm|audio|table|datarec) #REQUIRED</td><td>Refer to Appendix E for Attribute and Entity explanations.</td></tr>
<tr><td>descr CDATA #REQUIRED></td><td>Refer to Appendix E for Attribute and Entity explanations.</td></tr>
<tr><td><!ELEMENT tbl - - (no?,tt,tdim?,tby) -(fig|fn|tbl|ctbl)></td><td>The content model for simple tables (tbl) is composed of one model group. The elements in the content model of a simple table (tbl) must begin with a number (no), followed by a table title (tt), row/column dimensions (tdim), and table body. Both number (no) and row/column dimensions (tdim) are optional and occur only once. Table title (tt) and table body (tby) are required.
No figures (fig), footnotes (fn), simple (tbl) or complex (ctbl) tables are allowed within a simple table.</td></tr>
</table>

Table 4-1 -- continued.

<table>
<tr><th>Line in Model</th><th>Explanation of Content Model</th></tr>
<tr><td><!ELEMENT no - - (#PCDATA)></td><td>The content model for simple table number (no) consists of text (#PCDATA).</td></tr>
<tr><td><!ELEMENT tt - - (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*></td><td>The content model for simple table title (tt) is composed of three model groups. The simple table title (tt) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a geometric formula (f).
(4) a figure reference (fgr).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT tby - - (th*,tsb*,row*)></td><td>The content model for simple table body (tby) is composed of one model group. The simple table body (tbl) consists of a table column header (th), followed by a table column subordinate header (tsb), then table rows (row). All three elements are optional and repeatable.</td></tr>
<tr><td><!ELEMENT row - - (tsb?,c*)></td><td>The content model for table row (row) is composed of one model group. The table row (row) consists of a table stub line (tsb) followed by cells (c).</td></tr>
<tr><td><!ELEMENT th - - (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*></td><td>The content model for table column header (th) is composed of three model groups. The table column header (th) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a geometric formula (f).
(4) a figure reference (fgr).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT tsh - - (#PCDATA|((it|b|e1|e2|e3)|f|fgr))*></td><td>The content model for table column subordinate header (tsh) is composed of three model groups. The table column subordinate header (tsh) can consist of only one of these structural situations:
(1) text (#PCDATA).
(2) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(3) a geometric formula (f).
(4) a figure reference (fgr).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT tsb - - (p,(p|(l1|l2|l3)|f)*)></td><td>The content model for simple table stub line (tsb) is composed of three model groups. The simple table stub line (tsb) must begin with a paragraph (p). There are three structural possibilities that can occur following the required paragraph (p):
(1) another paragraph (p).
(2) only one of the following lists: bulleted (l1), unmarked (l2), or enumerated (l3).
(3) a geometric formula (f).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT c - - (p,(p|(l1|l2|l3)|f)*)></td><td>The content model for simple table cell (c) is composed of three model groups. The simple table cell (c) must begin with a paragraph (p). There are three structural possibilities that can occur following the required paragraph (p):
(1) another paragraph (p).
(2) only one of the following lists: bulleted (l1), unmarked (l2), or enumerated (l3).
(3) a geometric formula (f).
The entire content model is optional and repeatable.</td></tr>
</table>

Table 4-1 -- continued.

<table>
<tr><th>Line in Model</th><th>Explanation of Content Model</th></tr>
<tr><td><!ATTLIST c shaded (y|n) #REQUIRED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>topline NAME #REQUIRED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>botline NAME #REQUIRED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>lftline NAME #REQUIRED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>rgtline NAME #REQUIRED></td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td><!ELEMENT ctbl - - (cthd,ctby,ctbf) -(fig|fn|tbl|ctbl)></td><td>The content model for complex table (ctbl) is composed of one model group. The complex table (ctbl) begins with a complex table head (cthd), followed by the complex table body (ctby), and then the complex table foot (ctbf). All three elements are required. Figures (fig), footnotes (fn), simple (tbl) and complex (ctbl) are not allowed within a complex table.</td></tr>
<tr><td><!ELEMENT cthd - - (no?,ctt?,tdim?,cthr*)></td><td>The content model for complex table head (cthd) is composed of one model group. The complex table head (cthd) begins with a number (no), followed by a complex table title (ctt), row/column dimensions (tdim), and complex table header row (cthr). Number, complex table title, and row/column dimensions are optional and occur only once. Complex table header row is optional and repeatable.</td></tr>
<tr><td><!ELEMENT ctt - - ((it|b|e1|e2|e3)|f|#PCDATA)*></td><td>The content model for complex table title (ctt) is composed of two model groups. The complex table title (ctt) can consist of only one of these structural situations:
(1) one of the following: italics (it), bold (b), straight emphasis (e1), scientific name (e2), or reference title (e3).
(2) a geometric formula (f).
(3) text (#PCDATA).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT tdim - - (colwid,rowhgt)></td><td>The content model for table dimensions (tdim) is composed of one model group. The row/column dimensions (tdim) begin with column widths in inches (colwid), and are followed by row widths in inches (rowhgt). Both elements are required.</td></tr>
<tr><td><!ELEMENT colwid - - (#PCDATA)></td><td>The content model for column dimensions (colwid) consists of text. The column dimensions (colwid) are a comma-delimited set of column widths in 1200^{ths} of an inch. These dimensions are used to determine the span of cells across any number of columns. See Appendix G for more explanation.</td></tr>
<tr><td><!ELEMENT rowhgt - - (#PCDATA)></td><td>The content model for row dimensions (rowhgt) consists of text. The row dimensions (rowhgt) are a comma-delimited set of row heights in 1200^{ths} of an inch. These dimensions are used to determine the span of cells across any number of rows. See Appendix G for more explanation.</td></tr>
<tr><td><!ELEMENT cthr - - (cth*)></td><td>The content model for complex table header row (cthr) is composed of one model group. Each complex table header row (cthr) consists of an optional and repeatable complex table header (cth).</td></tr>
<tr><td><!ELEMENT cth - - (p|(l1|l2|l3)|f)*></td><td>The content model for complex table header (cth) is composed of two model groups. There are three structural possibilities that can occur within a complex table header (cth):
(1) a paragraph (p).
(2) only one of the following lists: bulleted (l1), unmarked (l2), or enumerated (l3).
(3) a geometric formula (f).
The entire content model is optional and repeatable.</td></tr>
</table>

Table 4-1 -- continued.

<table>
<tr><th>Line in Model</th><th>Explanation of Content Model</th></tr>
<tr><td><!ELEMENT ctsbl - - (p|(l1|l2|l3)|f)*></td><td>The content model for complex table stub line (ctsbl) is composed of two model groups. There are three structural possibilities that can occur within a complex table stub line (ctsbl):
(1) a paragraph (p).
(2) only one of the following lists: bulleted (l1), unmarked (l2), or enumerated (l3).
(3) a geometric formula (f).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT cte - - (p|(l1|l2|l3)|f)*></td><td>The content model for complex table entry (cte) is composed of two model groups. There are three structural possibilities that can occur within a complex table entry (cte):
(1) a paragraph (p).
(2) only one of the following lists: bulleted (l1), unmarked (l2), or enumerated (l3).
(3) a geometric formula (f).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ELEMENT ctc - - (p|(l1|l2|l3)|f)*></td><td>The content model for complex table caption (ctc) is composed of two model groups. There are three structural possibilities that can occur within a complex table identifier/caption(ctc):
(1) a paragraph (p).
(2) only one of the following lists: bulleted (l1), unmarked (l2), or enumerated (l3).
(3) a geometric formula (f).
The entire content model is optional and repeatable.</td></tr>
<tr><td><!ATTLIST cth align (l|c|r|d) #IMPLIED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>valign (t|m|b) #IMPLIED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>cb NUMBER #IMPLIED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>ce NUMBER #IMPLIED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>rb NUMBER #IMPLIED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>re NUMBER #IMPLIED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>shaded (y|n) #REQUIRED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>topline NAME #REQUIRED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>botline NAME #REQUIRED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>lftline NAME #REQUIRED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>rgtline NAME #REQUIRED></td><td>Refer to Appendix E for Attribute explanations.</td></tr>
</table>

Table 4-1 -- continued.

<table>
<tr><th>Line in Model</th><th>Explanation of Content Model</th></tr>
<tr><td><!ATTLIST ctsbl align (l|c|r|d) #IMPLIED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>valign (t|m|b) #IMPLIED></td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td><!ATTLIST ctc align (l|c|r|d) #IMPLIED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>valign (t|m|b) #IMPLIED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>cb NUMBER #IMPLIED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>ce NUMBER #IMPLIED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>rb NUMBER #IMPLIED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>re NUMBER #IMPLIED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>shaded (y|n) #REQUIRED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>topline NAME #REQUIRED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>botline NAME #REQUIRED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>lftline NAME #REQUIRED</td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td>rgtline NAME #REQUIRED></td><td>Refer to Appendix E for Attribute explanations.</td></tr>
<tr><td><!ELEMENT ctby - - (ctr)*></td><td>The content model for complex table body (ctby) is composed of one model group. The complex table body (ctby) consists of complex table rows (ctr), which are optional and repeatable.</td></tr>
<tr><td><!ELEMENT ctr - - (ctsbl?,ctc*)></td><td>The content model for complex table row (ctr) is composed of one model group. The complex table rows (ctr) begin with a complex table stub line (ctsbl) that is optional and can occur only once. Complex table entries (ctc) follow any complex table stub line (ctsbl). They are optional and repeatable.</td></tr>
<tr><td><!ELEMENT ctbf - - (ctc)></td><td>The content model for complex table foot (ctbf) is composed of one model group. The complex table foot (ctbl) consists of a complex table identifier/caption (ctc) that is required.</td></tr>
<tr><td colspan="2">See Appendix E for a description of the general entities that represent special characters and symbols in an SGML instance (e.g., an ampersand (&) would be represented as & in an instance). The retrieval system would replace any general entity (e.g., &) found in the instance with the actual character or symbol (e.g., &).</td></tr>
</table>

With the exception of hyperlink elements and column and row elements in simple and complex tables, the model is a subset of the AAP Article model. Having the FCES DTD as a subset of the AAP Article model provides three benefits for FCES publications. First, little customization is needed to convert the information in FCES publications to ANSI standard format. Second, the information in FCES publications has the benefit of terminology and names that are widely used in the United States. Third, the FCES model provides potential portability of FCES publications to systems that read information in AAP Article model format.

The basis for the design and development of the FCES model was to allow information in the FCES publications to be described in a form independent of any computer hardware and software. A process must be established that verifies the model is an application-independent storage of FCES publications.

CHAPTER V
MODEL VERIFICATION

FCES Publication Preparation and Conversion to SGML Format

Identification of Model Elements in FCES Publications

An electronic toolkit, known as FAST-WP, was developed at the University of Florida to make it easy for authors and word processors to add special codes to WordPerfect 5.1 (WordPerfect Corporation, 1993a) files running under DOS (Cilley and Watson, 1992a and 1992b). The special codes, WordPerfect styles with generic stylenames, were used to define structural areas within FCES publications. The styles were placed in FCES publications via a pop-up menu (Appendix F). FAST-WP was modified after the FCES DTD was developed to include generic styles that reflect the structural elements in the model. The author then used FAST-WP to apply generic styles to the subset of FCES publications. During this tagging process FAST-WP was tested, edited and updated as problems were encountered. Appendix G provides a table describing the relationships between the model elements and their generic styles that were placed in FCES publications.

Conversion of FCES Publications Into SGML Format

A computer program, WP2SGML (WordPerfect-to-SGML), was written to generate SGML instances from the FCES publications (Harrison et al., 1992). Currently, WP2SGML converts an FCES publication into an ASCII file with start and end tags that describe its structure. WP2SGML was written in C++, using object oriented features of the language. Each style has its own object for conversion from WordPerfect to SGML. The objects have inheritance so objects can be processed at a general or specific level. Instances (tagged FCES documents) generated by WP2SGML were validated for conformance to the FCES document model (DTD) using Exoterica's XGML Validator software (Version 1.0, 1991). Initially, some documents did not conform to the FCES model due to errors in the logic of WP2SGML, structural problems in the FCES model, or tagging errors by the author. Editing, testing, evaluating, and updating of the FCES DTD was done when any structural differences were found in the tagged FCES documents.

Many tags are placed with WordPerfect styles. Certain styles in WordPerfect, such as footnotes, figures, and tables are readily apparent in WordPerfect by the conversion program. The conversion program uses the codes embedded in the WordPerfect file to detect where, for example, a footnote begins and ends, and where it should be placed. It then places the footnote, along with start and end tags, into the FCES instance (ASCII tagged text file). An example of a more

difficult conversion is the paragraph, where the program requires the recognition of hard returns to correctly detect the beginning and ending of a paragraph. Once the beginning and end of a paragraph are detected, the software places start and end tags into the ASCII file. There are several structural elements that are difficult to detect in a standard WordPerfect document. These elements must be tagged with styles for identification so the conversion program can detect them. The pop-up menu is used to tag those structural elements, such as heading levels and hyperlinks.

Conversion of FCES Instances Into Retrieval Format

Each element was ranked for importance in on-screen display before the conversion of FCES instances into retrieval system format. The author used the rankings as a starting point for evaluating possible limitations a retrieval system might have delivering specific information in FCES publications. A retrieval software could be deemed inadequate when extremely important elements cannot be displayed on-screen. The initial ranking also provided the author a way to justify whether retrieval software can adequately display FCES information on-screen via different methods (e.g., pop-up box or full screen display of hyperlink material). The ranking levels for the elements were extremely important (A), somewhat important (B) or not needed for display (N). There were twenty two elements that were extremely important, thirty two

somewhat important and fourteen that were not needed for display.

Each retrieval system requires different processes to successfully translate FCES instances into a format it understands. A brief description of the processes for each retrieval system is as follows.

<u>Candide: The Semantic Data Modelling Language For FAIRS DISC8 And DISC9</u>

Query searches on tagged documents that are stored by themselves in a database are cumbersome and inefficient (Beck and Watson, 1992). Candide was designed specifically for storing both the structure and content (semantic) of FCES publications. Candide is a database management system that has storage, retrieval and query facilities. The semantic data model (Candide) is used to decompose each publication into objects (Beck et al., 1989a).

The objects represent the meaning of words and can interact with each other to represent complex data (Beck et al., 1989b). A document can be represented using these objects. Candide differs from other semantic data models by its uniform treatment of data objects, query objects and view objects. Candide also provides for query searches about the structural relationships between objects.

FAIRS CD-ROM DISC8

FCES instances are grouped together in handbooks or topics. A program called XTRAN (Exoterica, 1990) uses a set of rules to convert the SGML instances into an object-oriented format similar to PROLOG. The structure of each file (fname.out) determines their translation (CDM) into the Candide database (Beck et al., 1989a).

These Candide files are converted to ASCII files (fname.asc) using DB2ASC, where each individual object or chunk within a file was treated as a single ASCII file. This provides a set of ASCII chunks that link together through hyperlinks within the SGML instance. Editors then review the chunks for formatting errors and prepare the ASCII files (chunks) into a format for on-screen display. Tables represent a major problem encountered by editors and translation. The editors had to chunk all tables by hand because XTRAN did not have sufficient rules to support tables. Chunking involves breaking down a document into smaller pieces. Tables were extracted from the WordPerfect file and saved as an ASCII file. Table extraction allows simultaneous editing of tables and other chunks from an SGML instance. The final procedure is a two-step process (TXT2OBJ). The first step is translation of ASCII files (fname.asc) into text files. It removes all the hard returns and places special codes at the end of paragraphs. The conversion of the text files into objects, by retagging or conversion, is the last

step. After this conversion, the objects are placed into a retrievable database such as DISC8. The database includes both data files and an index. The index file is a table reference for each chunk of information on DISC8. DISC8 production relied on editors and chunking procedures for on-screen display of documents. The SGML elements in the FCES model were not used in the process.

A shell program (WP2DB) was written to create, write script, and run batch files (three) for placing WordPerfect files into a Candide database. Each step in the process has its own utility. A failure at any step stops the processing on that document. The first batch file reads WordPerfect files, then converts them into SGML format (fname.sgm) using WP2SGML. Tables were extracted and saved in ASCII format before this process. The shell then ensures each instance is a valid SGML document. The second batch file reads the SGML instances and creates the command line parameters for XTRAN. It runs XTRAN and converts each instance into objects (objects.out). The third batch file reads the fname.out file structure and initializes the parameters in object format for CDM. CDM translates the object files into Candide database format and places them in the database. Figure 5-1 describes the production process for DISC8.

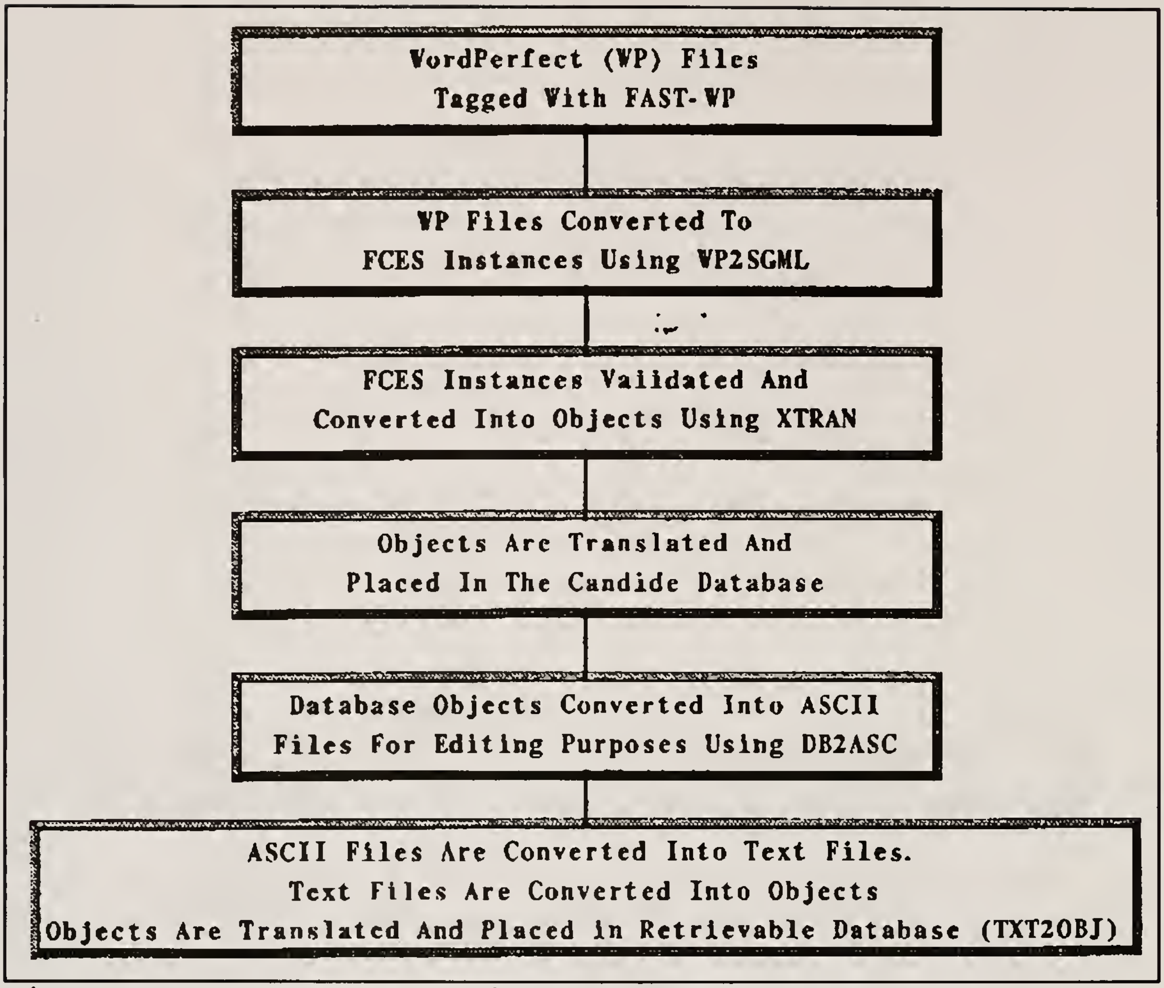


Figure 5-1 -- The production process for DISC8.

FAIRS CD-ROM DISC9

The significant difference in the development procedures for DISC8 and DISC9 was the use of the explicit structure in the FCES instances (SGML elements). Also, a parser was developed to make a one-step process out of the XTRAN and CDM steps. The parser or translator is written in C++, and uses a chart to translate elements into the Candide database. The

parser has one grammar rule for each SGML element. The grammar rules embody the way of analyzing the document that is being parsed. Along with the rule is a template with the Candide object for the specific SGML element. Each template is filled with the actual data that produced the object. The parser uses the grammar rules to translate the input string and place the data into a template. The template produces the data object that is put into the Candide database. The rules and templates in the chart parser show the elements and how they will appear in the Candide database. The grammar rules from the chart parser are also stored in the Candide database. Figure 5-2 describes the production process for DISC9.

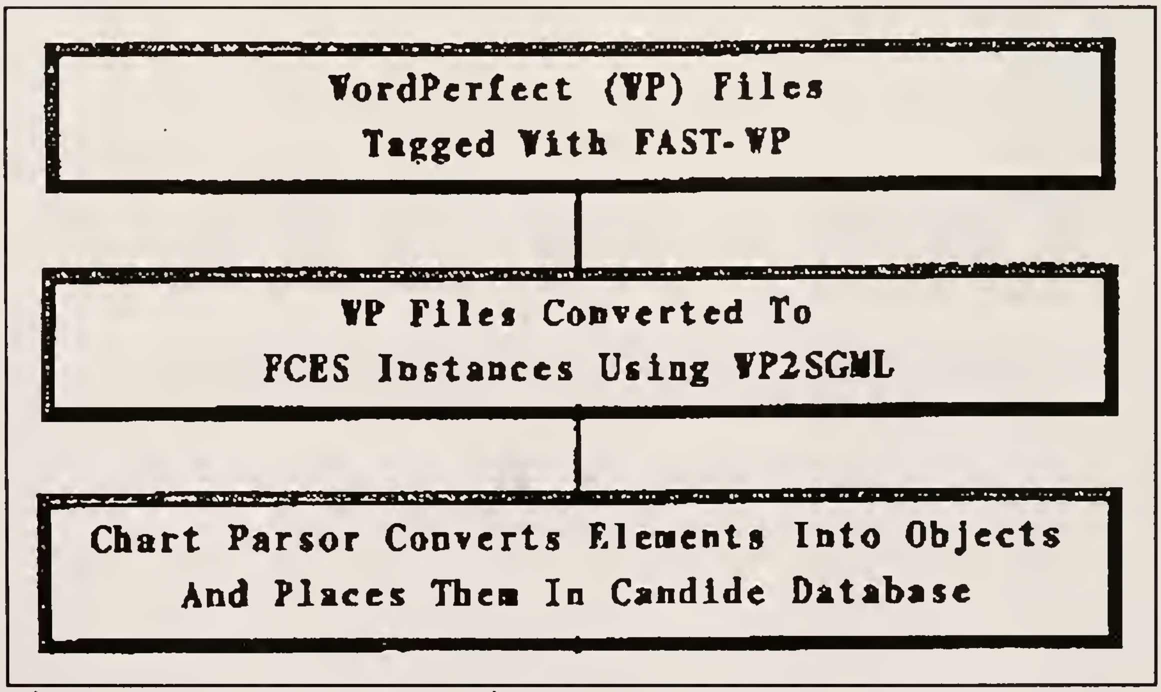


Figure 5-2 -- The production process for DISC9.

Multimedia Viewer

Microsoft Multimedia Viewer (1994), is a framed-based retrieval software that reads files in Rich Text Format (RTF). All files created from SGML instances must be saved in RTF files (ie., topic files). RTF statements, which are specially formatted tags that specify a particular type of formatting information, are presented in these topic files. Multiple topic files are grouped together in the Viewer project file. Other elements in a topic file include destinations, control symbols, and groups. Both font and color tables can be developed to define topic information. The key process here is that Multimedia Viewer provides the formatting, while the user provides the structure. A WordPerfect macro (SGML2RTF) was written to automate the process from SGML instance to Multimedia format. The process is as follows:

1) Put all instances and graphics to be converted into a directory.
2) Select that directory for conversion.
3) Read "factshts.tag" file for elements to be considered.
4) Set global parameters.
5) Retrieve footnote elements.
6) Copy footnote elements to utility document.
7) Copy all sections <sec> to utility document.
8) Create 1 frame per document level (section <sec>, subsection level 1 <ss1>, subsection level 2 <ss2>, and subsection level 3 <ss3>).

9) Format any elements (e.g., lists, topic paragraphs, paragraphs) in each section.
10) Write out rtf file.
11) Generate the table of contents entry for the current instance.
12) Next instance goes through the conversion process.
13) The macro ends.
14) Use Multimedia Viewer to create the fname.mvb file from the project file, graphic files, and RTF files.

<u>Guide</u>

Guide (InfoAccess, 1994) is an electronic publishing system that is designed around an object-oriented information model. The model presents information as a series of linked objects and manages relationships between them. All document components, from a single word to a graphic, can be represented as an object. Once each object is defined by a command, it can be linked to other objects. Guide provides for live or hot objects to be activated with the mouse using reference, expansion, note, and command buttons. Guide provides a scripting language (LOGiiX) to write definitions for command buttons. Guide differs from Multimedia viewer in that it provides the structure for on-screen display, while the user provides how it will be displayed (formatting). A program was written in the C programming language to convert

the SGML instances into HML (hypertext media language) format. The steps in the program are as follows:

1) Read and parse an instance (fname.sgm).
2) The first pass through each instance picks up information necessary for the conversion, as well as determining if any tables are present.
3) Convert any elements in instance to HML.
4) Write out the HML file.

AFter conversion to HML, the instances are then converted into Guide (fname.hml to fname.gui) format as follows:

1) Develop style file for formatting purposes.
2) Run Guide Writer on the instance converted to HML format (fname.hml).
3) Develop style file for tables. Files with a TMF extension are generated for each table.
4) Table Viewer software takes each fname.tmf (text) as input and displays the table on-screen (runtime).

Results of FCES Instances Converted to Retrieval Format

Converting FCES Instances into FAIRS Retrieval Format

Table 5-1 provides the evaluation of converting FCES instances into FAIRS retrieval format for DISC8 and DISC9. The DISC8 process relied on editors and chunking procedures to manually prepare the information for on-screen display. There was no automation in the production process. The DISC9 process made use of the explicit structure in the FCES

instances. All the elements were automatically translated to the FAIRS retrieval format because the chart translator creates a rule and template with an object for each element in the model. The objects were then translated and placed into the FAIRS retrievable database. The translation of the objects and rules directly into the retrievable database is an automatic process for all elements in the FCES model. Table 5-2 shows that the translation of elements into DISC8 format was a manual process, while the translation to DISC9 format was a totally automated process.

Converting FCES Instances into Multimedia Viewer Format

Table 5-1 provides some translation comments on the conversion of FCES instances to Multimedia Viewer format. The process from FCES instances to Multimedia Viewer format was primarily an automated process. Table 5-2 shows that superscript and subscript were the only elements that Multimedia Viewer could not translate to retrieval system format.

Converting FCES Instances into Guide Format

Table 5-1 provides some translation comments on the conversion of FCES instances to Guide format, which was an automated process. Table 5-2 shows that Guide could translate all elements in the model to retrieval system format.

Table 5-1 -- Priority Level Of Elements For On-screen Display and The Level Of Difficulty Retrieval Software Must Go Through To Use Them.

Element Name	Priority Level (1)	Difficulty Level of Retrieval Software to Translate Elements (2)				Translation Comments
		FAIRS		MultiMedia Viewer (MMV)	Guide	
		disc8	disc9			
article	N	2	1	1	1	Ignored in MMV. Replace article with "HML" element for Guide.
fm	N	2	1	1	1	Ignored in MMV and Guide.
tig	N	2	1	1	1	Ignored in MMV and Guide.
atl	A	2	1	1	1	Formatted as a pop-up menu in MMV. Title is formatted in Guide.
it	B	2	1	1	1	Rich Text Format (RTF) codes replace this tag in MMV. Replaced with a style override tag (SOT) in Guide.
b	B	2	1	1	1	Rich Text Format (RTF) codes replace this tag in MMV. Replaced with a SOT in Guide.
e1	N	3	3	3	3	Not available for translation.
e2	B	2	1	1	1	A macro checks to see if anything is inside of <e2>. If not, then the text is tag in italics (MMV). Element is in italics in Guide.
e3	B	2	1	1	1	<e3> is always ignored because of the italics tag inside of it (MMV). In italics in Guide.
fgr	A	3	3	3	3	Not available for translation.
f	N	2	1	1	1	Ignored in MMV and Guide.
sup	B	2	1	2	1	No translation to MMV. Replaced with a SOT in Guide.
inf	B	2	1	2	1	No translation to MMV. Replaced with a SOT in Guide.
au	N	2	1	1	1	Ignored in MMV and Guide.
onm	A	3	3	3	3	Not available for translation.
snm	A	2	1	1	1	Used as a hyperlink to the author information in MMV. Not needed in Guide.
pubfm	N	2	1	1	1	Ignored in MMV and Guide.
aid	A	2	1	1	1	Used as a hyperlink to the title information in MMV. In Guide, translate it to a note button. aid would open title footnote information.
issn	B	3	3	3	3	Not available for translation.
avl	N	3	3	3	3	Not available for translation.

Table 5-1 -- continued.

Element Name	Priority Level (1)	Difficulty Level of Retrieval Software to Translate Elements (2)				Translation Comments
		FAIRS		MultiMedia Viewer (MMV)	Guide	
		disc8	disc9			
crt	N	2	1	1	1	Ignored in MMV and Guide.
crd	B	2	1	1	1	Ignored in MMV and Guide.
abs	A	2	1	1	1	Ignored in MMV.
mo	N	3	3	3	3	Not available for translation.
day	N	3	3	3	3	Not available for translation.
yr	A	2	1	1	1	Formatted with RTF and right justified in MMV. Formatted also in Guide.
h	A	2	1	1	1	Made as bold text (MMV). Replaced with SOT in Guide.
bdy	N	2	1	1	1	Ignored in MMV. In Guide, it translates into 1 frame tag, which has many expansions.
sec	A	2	1	1	1	Used to build the bookmarks that separates structural levels in publications (MMV). Used to format the expansion button tag "st".
ss1	A	2	1	1	1	Used to build the bookmarks that separates structural levels in publications (MMV). Used to format the expansion button tag "st" in Guide.
ss2	A	2	1	1	1	Used to build the bookmarks that separates structural levels in publications (MMV). Used to format the expansion button tag "st" in Guide.
ss3	A	2	1	1	1	Used to build the bookmarks that separates structural levels in publications (MMV). Used to format the expansion button tag "st" in Guide.
st	A	2	1	1	1	The hyperlink to sec, ss1, ss2 and ss3 in MMV. Expansion button tag for sec, ss1, ss2 and ss3 in Guide.
p	A	2	1	1	1	Starts a new line for text in MMV and Guide.
topl	A	2	1	1	1	Currently treated as text MMV and Guide.
l1	A	2	1	1	1	L1 is replaced by bookmarks to format the bulleted list (MMV). Used to determine the type of formatting for the items in Guide.
l2	A	2	1	1	1	L2 is replaced by bookmarks to format the unmarked list (MMV). Used to determine the type of formatting for the items in Guide.
l3	A	2	1	1	1	L1 is replaced by bookmarks to format the enumerated list (MMV). Used to determine the type of formatting for the items in Guide.
lh	B	2	1	1	1	Hyperlink to l1, l2 or l3 if there is one in MMV and Guide.

Table 5-1 -- continued.

Element Name	Priority Level (1)	Difficulty Level of Retrieval Software to Translate Elements (2)				Translation Comments
		FAIRS		MultiMedia Viewer (MMV)	Guide	
		disc8	disc9			
li	A	2	1	1	1	Items are formatted based on the type of list in MMV and Guide. Each list has an indent, followed by an asterisk or number, then a tab. Unmarked lists only have an indent.
fig	A	2	1	*	*	Not available for translation.
fn	B	2	1	1	1	Information displayed as hypertext for aid and snm (MMV). A location tag for text in footnotes.
hyp	N	2	1	1	1	Initial pass sets up lword and act in an array in MMV and Guide.
lword	A	2	1	1	1	Reference in text for hyperlink (MMV). A command button for Guide Image Viewer.
act	A	2	1	1	1	A pop-up menu (MMV). In Guide, the filename acts as a graphic to send to image viewer. Notebuttons are used for text and tables. Executable programs are sent to a launch program.
tbl	N	2	1	1	1	Places bookmarks at beginning and end of simple tables (MMV). Translates to "table" in TMF.
no	B	3	3	3	3	Not available for translation.
tt	B	3	3	3	3	Not available for translation.
tby	B	2	1	1	1	Used as a marker where table body begins and headers end (MMV). Translates to "tbody" in TMF for Guide.
row	B	2	1	1	1	Tells software when to begin and end a row (MMV). Translates to "row" in TMF for Guide.
th	B	3	3	3	3	Not available for translation.
tsh	B	3	3	3	3	Not available for translation.
tsb	B	3	3	3	3	Not available for translation.
c	B	2	1	1	1	Text within a table (MMV). Translates to "cell" in TMF for Guide.
ctbl	B	2	1	1	1	Places bookmarks at beginning and end of complex tables (MMV). Translates to "table" in TMF for Guide.
cthd	B	2	1	1	1	Ignored in MMV. Used as a placeholder in Guide.
ctt	B	3	3	3	3	Not available for translation.
tdim	B	2	1	1	1	Ignored in MMV.

Table 5-1 -- continued.

Element Name	Priority Level (1)	Difficulty Level of Retrieval Software to Translate Elements (2)				Translation Comments
		FAIRS		MultiMedia Viewer (MMV)	Guide	
		disc8	disc9			
colwid	B	2	1	1	1	Locates the first column width numbers and processes them all (MMV). Numbers are stored for later use by element cte in Guide.
rowhgt	B	2	1	1	1	Locates the first row height numbers and processes them all (MMV). Numbers are stored for later use by element cte in Guide.
cthr	B	2	1	1	1	Becomes a regular row (MMV). Nonscrolling element at top of table in Guide.
cth	B	2	1	1	1	Treated as a cell MMV and guide.
ctsbl	B	3	3	3	3	Not available for translation.
cte	B	2	1	1	1	Used to set up cells and determine column and row span (MMV). Translates to "cell" in TMF for Guide.
ctc	B	3	3	3	3	Not available for translation.
ctby	B	2	1	1	1	Used as a marker where table body begins and headers end (MMV). Translates to "tbody" in TMF for Guide.
ctr	B	2	1	1	1	Each end tag determines the start of a new row. Translates to "row" in TMF for Guide.
ctbf	B	3	3	3	3	Not available for translation.

1) Elements are ranked for on-screen display as extremely important (A), Somewhat important (B) or not needed for display (N).
2) Translation from the elements to the format of the retrieval system can be either automatic (1), manual (2) or presently not in conversion process to SGML format (3).

Table 5-2 -- Summary of Element Translation to Retrieval System Format.

On-screen Priority Level	Number of Elements in FCES Model that Convert to Each Retrieval System Format Either Automatically, Manually or Not Applicable Due To No Translation of Element to SGML Format.											
	FAIRS DISC8			FAIRS DISC9			MultiMedia Viewer			Guide		
	Automatic	Manual	Not Applicable	Automatic	Manual	Not Applicable	Automatic	Manual	Not Applicable	Automatic	Manual	Not Applicable
A	0	19	3	19	0	3	19	0	3	19	0	3
B	0	22	10	22	0	10	20	2	10	22	0	10
N	0	10	4	10	0	4	10	0	4	10	0	4

Interpretation of Elements by Automated Retrieval Systems

Software was written to automate the translation of elements into each retrieval system format (DISC9, Multimedia Viewer and Guide) for on-screen display. All of the extremely important elements (A) were translated to the three retrieval systems. All of the somewhat important elements (B) were translated to DISC9 and Guide retrieval systems. The Mulitmedia Viewer retrieval system can not display superscript or subscript characters.

Possible Model Changes

Simplification of the model can reduce both the knowledge authors need to use FAST-WP and redundancy of elements in FCES instances. There are many elements that "ride" along in multiple content models as holders for other elements. Some of these elements might include front matter (fm), title group (tig), figure reference (fgr), geometric formula (f), author (au), publishers front matter group (pubfm), copyright notice (crt), copyright notice date (crd), complex table head (cthd), row/column dimensions for simple and complex tables (tdim), and complex table header (cth). The content models for each of these elements could be represented by themselves in the model.

CHAPTER VI
SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

Summary

The objective of this research was to model the structure and represent the information in FCES publications in an electronic form that is independent of any specific computer hardware or software. The initial research began with a graduate course in SGML principles and practices that provided a starting point for the author's document analysis on FCES publications. Document analysis proved to be an ongoing process throughout the research. The class also illustrated the difficulty of attaining agreement on a model for a set of FCES publications from two or more groups. To forestall possible disagreements on FCES document structure, the author proposed to adopt an outside standard rather than develop an in-house model. After review of current American publication models, coupled with the author's document analysis, the author selected the Association of American Publisher's (AAP) Article model as the best representation of the structure in FCES publications. The author initially developed a subset of the AAP Article model, and later added structural elements unique to FCES publications (Appendix E). Removal of the

unique elements would produce a model compatible with the AAP Article DTD.

An in-house publishing tool (FAST-WP) was based, in part, on the FCES model developed by the author. The generic styles in FAST-WP reflect the structure of the FCES model. The author used FAST-WP as the publishing tool to tag the subset of FCES publications.

Once the FCES publications were tagged, an in-house software (WP2SGML) was developed to convert each publication into SGML format (instance). The author helped develop the structural conversions that WP2SGML uses in the conversion process. These structural conversions were based on the FCES model. The author then used a parser to verify that each FCES instance conformed to the ISO 8879-1986 standards and the FCES model. This verification proved that the FCES model describes the content of FCES publications.

After conversion of FCES tagged publications into instances, the author ranked the priority level of each element in the FCES model for on-screen display. The author then developed how the elements in the FCES model would appear on-screen. Software was then written to convert FCES instances into Multimedia Viewer and Guide retrieval system format. All elements in the FCES model were automatically translated to FAIRS DISC9 and Guide retrieval system format. All elements in the FCES model, except superscript and subscript, were automatically translated to Multimedia Viewer

format. The automatic translation of FCES publications in SGML format to FAIRS DISC9, Multimedia Viewer and Guide retrieval formats verify that the FCES model is application-independent.

Conclusions/Findings

After converting a set of FCES publications into several retrieval system formats the following conclusions were drawn:

1) SGML was a suitable method of rules and syntax for modeling FCES publications.
2) The FCES model was validated based on ISO standard 8879-1986.
3) The model provided the structure for automatic conversion of FCES instances into retrieval system format.
4) The model was verified as a suitable way of describing information in FCES publications by the automatic conversion of FCES instances into retrieval system format.

Observations

Other findings that were not part of the research project, but observed during the process include the following:

1) Document analysis is an ongoing process that affects the entire conversion process. It determines document

preparation time and the degree of automation in the conversion process.

2) Prior to development of the DTD, decide who has the authority to change the SGML model (DTD). Application-independent document storage revolves around the DTD. The DTD is the single most important aspect of any SGML project. Previously converted publications must be revalidated when there are any changes in the model.

3) When developing the FCES model, the more rigid the structure the easier the implementation.

4) Use an existing model as a starting point to describe the structure of publications. A subset of an existing model provides greater access to the information.

5) Eliminate any "rider" elements that tag along with other elements. If possible, use only one element to describe the information.

6) Avoid recursive content models to simplify automated conversion from word processing files to SGML format (instance).

7) The specific retrieval software(s) used is not important.

8) A high priority when selecting a retrieval system should be what feature(s) are required for interpreting information in an instance.

9) Additional conversion time is required when a retrieval system requires FCES instances in a proprietary SGML model format.

Recommendations

1) Continue document analysis of FCES publications, refining structural properties to decrease the author's knowledge requirement of the process.
2) Update model as publications change, ensuring compatibility between current and older FCES publications.
3) Generate reports when there are any changes or revisions in the model.
4) Design new templates to aid authors in the tagging process. An example is a document template that has blanks to fill in the front matter of a publication.
5) Develop tools to automatically hyperlink references in FCES publications for tables and graphics.
6) Continue reviewing retrieval software as new applications become available.
7) Link multiple software such as authoring, SGML conversion, error checking and validation, and retrieval software into a seamless interface tool for the author.
8) Provide on-line documentation as a help facility for entire document conversion process.
9) Support a wider variety of word processors.
10) Minimize or eliminate as much of the manual tagging process as possible.

GLOSSARY

The ISO 8879-1986 International Standard gives the following definitions to certain names or titles described in this research project:

abstract syntax (of SGML): Rules that define how markup is added to the data of a document, without regard to the specific characters used to represent the markup.

application: Text processing application.

attribute (of an element): A characteristic quality, other than type or content.

attribute definition: A member of an attribute definition list; it defines an attribute name, allowed values, and default value.

attribute definition list: A set of one or more attribute definitions defined by the attribute definition list parameter of an attribute definition list declaration.

base document element: A document element whose document type is the base document type.

base document type: The document type specified by the first document type declaration in a prolog.

CDATA: Character data.

CDATA entity: Character data entity.

character: An atom of information with an individual meaning, defined by a character repertoire.

comment: A portion of a markup declaration that contains explanations or remarks intended to aid persons working with the document.

concrete syntax (of SGML): A binding of the abstract syntax to particular delimiter characters, quantities, markup declaration names, etc..

conforming SGML application: An SGML application that requires documents to be conforming SGML documents, and whose documentation meets the requirements of this International Standard.

conforming SGML document: An SGML document that complies with all provisions of this International Standard.

content: Characters that occur between the start-tag and end-tag of an element in a document instance. They can be interpreted as data, proper subelements, included subelements, other markup, or a mixture of them.

NOTE - if an element has an explicit content reference, or its declared content is "EMPTY", the content is empty. In such cases, the application itself may generate data and process it as though it were content data.

(content) model: Parameter of an element declaration that specifies the model group and exceptions that define the allowed content of the element.

core concrete syntax: A variant. of the reference concrete syntax that has no short reference delimiters.

data: The characters of a document that represent the inherent information content; characters that are not recognized as markup.

data content: The portion of an element's content that is data rather than markup or a subelement.

data tag: A string that conforms to the data tag pattern of an open element. It serves both as the end-tag of the open element and as character data in the element that contains it.

declaration: Markup declaration.

declaration subset: A delimited portion of a markup declaration in which other declarations can occur.

NOTE - Declaration subsets occur only in document type, link type, and marked section declarations.

delimiter characters: Character class that consists of each SGML character, other than a name character or function character, that occurs in a string assigned to a delimiter role by the concrete syntax.

delimiter set: A set of assignments of delimiter strings to the abstract syntax delimiter roles.

delimiter (string): A character string assigned to a delimiter role by the concrete syntax.

descriptive markup: Markup that describes the structure and other attributes of a document in a non-system-specific manner, independently of any processing that may be performed

on it. In particular, it uses tags to express the element structure.

document: A collection of information that is processed as a unit. A document is classified as being of a particular document type.

NOTE - in this international Standard, the term almost invariably means (without loss of accuracy) an SGML document.

document character set: The character set used for all markup in an SGML document, and initially (at least) for data.

NOTE - When a document is interchanged between systems, its character set is translated to the receiving system character set.

document element: The element that is the outermost element of an instance of a document type; that is, the element whose generic identifier is the document type name.

document instance: Instance of a document type.

document type: A class of documents having similar characteristics; for example, journal, article, technical manual, or memo.

(document) type declaration: A markup declaration that contains the formal specification of a document type definition.

document (type) definition: Rules, determined by an application, that apply SGML to the markup of documents of a particular type. A document type definition includes a formal specification, expressed in a document type declaration, of

the element types, element relationships and attributes, and references that can be represented by markup. It thereby defines the vocabulary of the markup for which SGML defines the syntax.

NOTE - A document type definition can also include comments that describe the semantics of elements and attributes, and any application conventions.

DTD: Document type definition.

element: A component of the hierarchical structure defined by a document type definition. It is identified in a document instance by descriptive markup, usually a start-tag and end-tag.

NOTE - An element is classified as being of a particular element type.

element declaration: A markup declaration that contains the formal specification of the part of an element type definition that deals with the content and markup minimization.

element structure: The organization of a document into hierarchies of elements, with each hierarchy conforming to a different document type definition.

element type: A class of elements having similar characteristics; for example, paragraph, chapter, abstract, footnote, or bibliography.

entity: A collection of characters that can be referenced as a unit.

NOTES

1) Objects such as book chapters written by different authors, pi characters, or photographs, are often best managed by maintaining them as individual entities.

2) The physical organization of entities is system-specific, and could take the form of files, members of a partitioned data set, components of a data structure, or entries in a symbol table.

entity declaration: A markup declaration that assigns an SGML name to an entity so that it can be referenced.

entity reference: A reference that is replaced by an entity.

NOTE - There are two kinds: named entity and short reference.

entity set: A set of entity declarations that are used together.

NOTE - An entity set can be public text.

exclusions: Elements that are not allowed anywhere in the content of an element or its subelements even though the applicable content model or inclusions would permit them optionally.

general entity: An entity that can be referenced from within the content of an element or an attribute value literal.

generic Identifier: A name that identifies the element type of an element.

GI: Generic identifier.

ID: Unique identifier.

Inclusions: Elements that are allowed anywhere in the content of an element or its subelements even though the applicable model does not permit them.

instance (of a document type): The data and markup for a hierarchy of elements that conforms to a document type definition.

mark up: To add markup to a document.

markup: Text that is added to the data of a document in order to convey information about it.

NOTE - There are four kinds of markup: descriptive markup (tags), references, markup declarations, and processing instructions.

(markup) declaration: Markup that controls how other markup of a document is to be interpreted.

NOTE - There are 13 kinds: SGML, entity, element, attribute definition list, notation, document type, link type, link set, link use, marked section, short reference mapping, short reference use, and comment.

(markup) minimization feature: A feature of SGML that allows markup to be minimized by shortening or omitting tags, or shortening entity references.

NOTE - Markup minimization features do not affect the document type definition, so a minimized document can be sent to a system that does not support these features by first restoring the omitted markup. There are five kinds: SHORTTAG, OMITTAG, SHORTREF, DATATAG, and RANK.

minimization feature: Markup minimization feature.

model: Content model.

model group: A component of a content model that specifies the order of occurrence of elements and character strings in an element's content, as modified by exceptions specified in the content model of the element and in the content models of other open elements.

name: A name token whose first character is a name start character.

name character: A character that can occur in a name: name start characters, digits, and others designated by the concrete syntax.

name group: A group whose tokens are required to be names.

number: A name token consisting solely of digits.

parameter entity: An entity that can be referenced from a markup declaration parameter.

parameter entity reference: A named entity reference to a parameter entity.

parsed character data: Zero or more characters that occur in a context in which text is parsed and markup is recognized. They are classified as data characters because they were not recognized as markup during parsing.

PCDATA: Parsed character data.

reference: Markup that is replaced by other text, either an entity or a single character.

reference concrete syntax: A concrete syntax, defined in this International Standard, that is used in all SGML declarations.

SGML: Standard Generalized Markup Language

SGML application: Rules that apply SGML to a text processing application. An SGML application includes a formal specification of the markup constructs used in the application, expressed in SGML. It can also include a nonSGML definition of semantics, application conventions, and/or processing.

NOTES

1) The formal specification of an SGML application normally includes document type definitions, data content notations, and entity sets, and possibly a concrete syntax or capacity set. If processing is defined by the application, the formal specification could also include link process definitions.

2) The formal specification of an SGML application constitutes the common portions of the documents processed by the application. These common portions are frequently made available as public text.

3) The formal specification is usually accompanied by comments and/or documentation that explains the semantics, application conventions, and processing specifications of the application.

4) An SGML application exists independently of any implementation. However, if processing is defined by the

application, the non-SGML definition could include application procedures, implemented in a programming or text processing language.

SGML character: A character that is permitted in an SGML entity.

SGML declaration: A markup declaration that specifies the character set, concrete syntax, optional features, and capacity requirements of a document's markup. It applies to all of the SGML entities of a document.

SGML document: A document that is represented as a sequence of characters, organized physically into an entity structure and logically into an element structure, essentially as described in this International Standard. An SGML document consists of data characters, which represent its information content, and markup characters, which represent the structure of the data and other information useful for processing it. In particular, the markup describes at least one document type definition, and an instance of a structure conforming to the definition.

SGML entity: An entity whose characters are interpreted as markup or data in accordance with this International Standard.

NOTE - There are three types of SGML entity: SGML document entity, SGML subdocument entity, and SGML text entity.

SGML parser: A program (or portion of a program or a combination of programs) that recognizes markup in conforming SGML documents.

NOTE - if an analogy were to be drawn to programming language processors, an SGML parser would be said to perform the functions of both a lexical analyzer and a parser with respect to SGML documents.

Standard Generalized Markup Language: A language for document representation that formalizes markup and frees it of system and processing dependencies.

start-tag: Descriptive markup that identifies the start of an element and specifies its generic identifier and attributes.

tag: Descriptive markup.

NOTE - There are two kinds: start-tag and end-tag.

text: Characters.

NOTE - The characters could have their normal character set meaning, or they could be interpreted in accordance with a data content notation as the representation of graphics, images, etc.

validating SGML parser: A conforming SGML parser that can find and report a reportable markup error if (and only if) one exists.

APPENDIX A
DEVELOPMENT OF AN SGML MODEL

This appendix describes the process for developing an SGML application. Given a subset of a class of documents such as fact sheets (Figure A-1), the documents are broken down into pieces (Figure A-2). A tree structure (Figure A-3) is developed describing the structure of the set of documents. The model (DTD) (Figure A-4), a vocabulary representation of the document structure, is written upon completion of the document analysis and validated for conformance to ISO 8879-1986 standards and SGML syntax. Tagged documents (Figure A-5), known as instances, are validated to ensure conformance (i.e., no errors) with the model (DTD).

ANGELICA

James M. Stephens

INTRODUCTION

Angelica is a European perennial plant sometimes grown in this country as a culinary herb. This member of the parsley family, related to carrots, grows in fields and damp places from Labrador to Delaware and west to Minnesota. Syria is believed to be its point of origin. One species (A. sylvestris) is called "Holy Ghost."

Use

The fresh stems and leafstalks are used as garnish and for making candied angelica. The seeds and the oil distilled from them are used in flavoring foods, and the aromatic roots are used in medicine. People in the north, particularly the Lapps, use it as a foodstuff, condiment, or medicine, and even chew it like tobacco.

Description

The robust growing angelica plant is 5-6 feet tall and resembles wild carrot, although the leaves are much broader. It has large petioles and a purple-colored root. Leaves are compound and flowers are borne in umbels like the carrot. It is a perennial plant that flowers every 2 years.

Culture

The plant thrives best in a moderately cool climate in semishade; therefore, it is unlikely to grow well in Florida. The plant is most readily propagated from division of old roots, which can be set either in the fall or spring about 18 inches apart in 3-foot rows. If seeds can be obtained, start seedlings in a seed bed. Then transplant to the garden.

Harvesting. Roots, stems, and seeds are harvested and used as needed, with some parts being ready 3-4 months after planting. Sometimes the roots of the first year plants are dug, but usually the harvest of roots is deferred until fall of the second year.

Figure A-1 -- An FCES Fact Sheet.

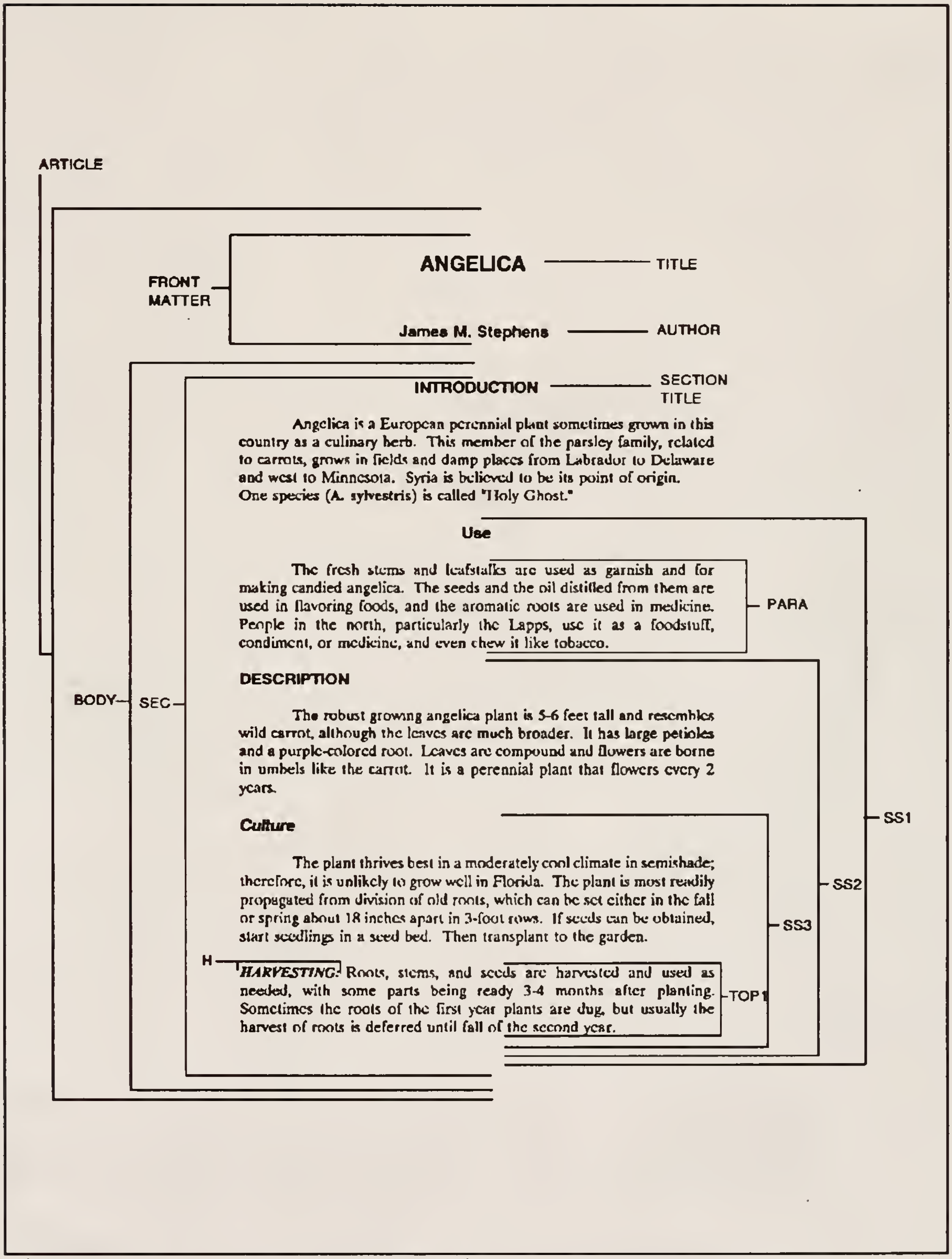


Figure A-2 -- Document Analysis of FCES Fact Sheet.

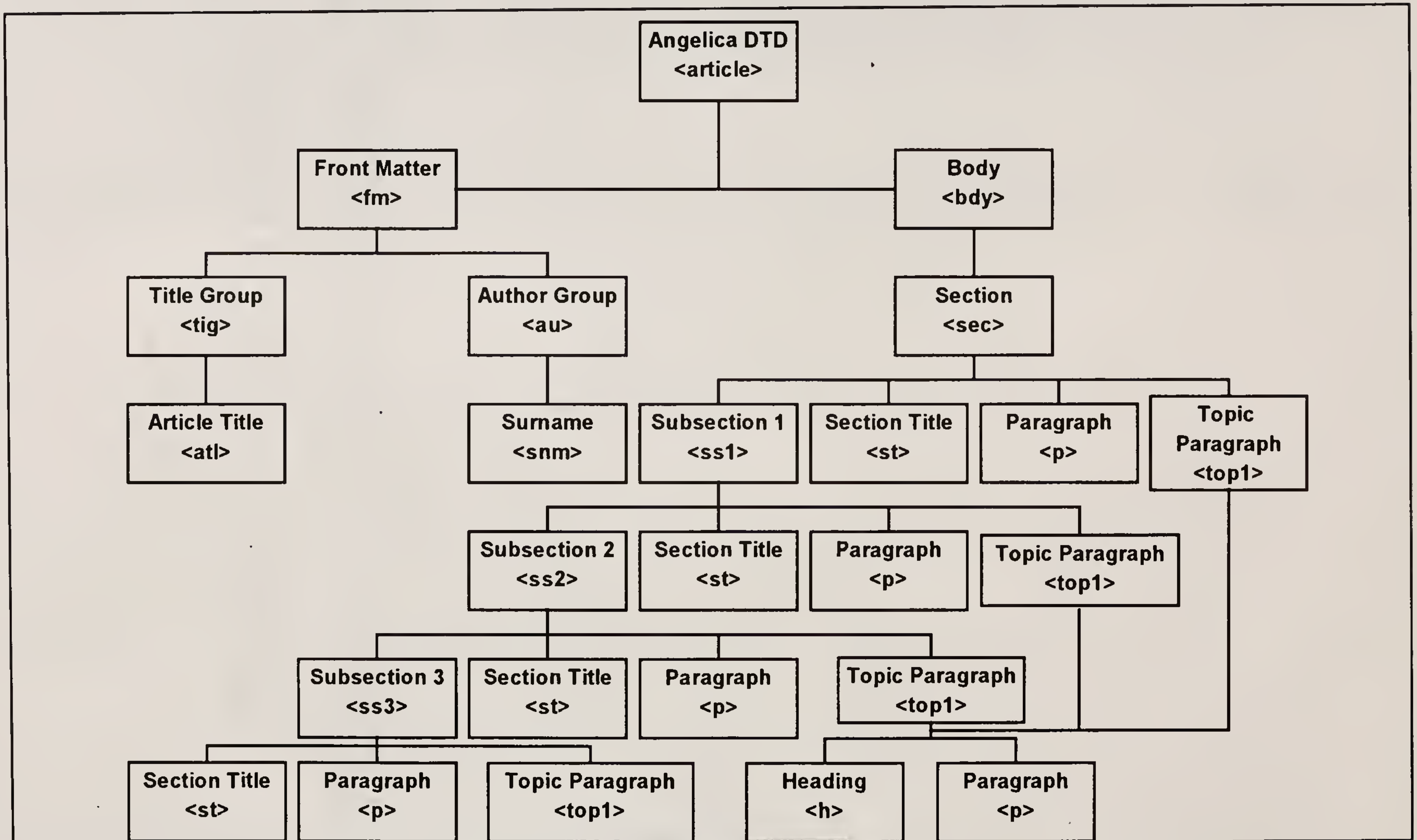


Figure A-3 -- Tree Structure of FCES Fact Sheet.

```
<!DOCTYPE article
[
<!ELEMENT article - -    (fm, bdy)>
<!ELEMENT fm      - -    (tig, au*)>
<!ELEMENT tig     - -    (atl)>
<!ELEMENT atl     - -    (#PCDATA)>
<!ELEMENT au      - -    (snm)>
<!ELEMENT snm     - -    (#PCDATA)>
<!ELEMENT bdy     - -    (sec)+>
<!ELEMENT sec     - -    (st, (p|top1)*, ss1*)>
<!ELEMENT ss1     - -    (st, (p|top1)*, ss2*)>
<!ELEMENT ss2     - -    (st, (p|top1)*, ss3*)>
<!ELEMENT ss3     - -    (st, (p|top1)*)>
<!ELEMENT top1    - -    (h?, p+)>
<!ELEMENT h       - -    (#PCDATA)>
<!ELEMENT st      - -    (#PCDATA)>
<!ELEMENT p       - -    (#PCDATA)>
]>
```

Figure A-4 -- DTD of FCES Fact Sheet.

<article><fm>
<tig><atl>ANGELICA</atl></tig>
<au><snm>James M. Stephens</snm></au></fm>
<bdy><sec><st>INTRODUCTION</st>
<p>Angelica is a European perennial plant sometimes grown in this country as a culinary herb. This member of the parsley family, related to carrots, grows in fields and damp places from Labrador to Delaware and west to Minnesota. Syria is believed to be its point of origin. One species (A. sylvestris) is called "Holy Ghost."</p>
<ss1><st>Use</st>
<p>The fresh stems and leafstalks are used as garnish and for making candied angelica. The seeds and the oil distilled from them are used in flavoring foods, and the aromatic roots are used in medicine. People in the north, particularly the Lapps, use it as a foodstuff, condiment, or medicine, and even chew it like tobacco.</p>
<ss2><st>Description</st>
<p>The robust growing angelica plant is 5-6 feet tall and resembles wild carrot, although the leaves are much broader. It has large petioles and a purple-colored root. Leaves are compound and flowers are borne in umbels like the carrot. It is a perennial plant that flowers every 2 years.</p>
<ss3><st>Culture</st>
<p>The plant thrives best in a moderately cool climate in semishade; therefore, it is unlikely to grow well in Florida. The plant is most readily propagated from division of old roots, which can be set either in the fall or spring about 18 inches apart in 3-foot rows. If seeds can be obtained, start seedlings in a seed bed. Then transplant to the garden.</p>
<top1><h>Harvesting.</h> <p>Roots, stems, and seeds are harvested and used as needed, with some parts being ready 3-4 months after planting. Sometimes the roots of the first year plants are dug, but usually the harvest of roots is deferred until fall of the second year. </p></top1>
</ss3></ss2></ss1></sec></bdy></article>

Figure A-5 -- Instance of FCES Fact Sheet.

APPENDIX B
PROCESS AND METHODOLOGY FOR DEVELOPING AN SGML APPLICATION

For the purposes of presenting a general procedure to develop an SGML application, the following steps were drawn from a basic SGML tutorial attended by the author (Graphic Communications Association, 1991). .

Recognizing SGML Applications

The first step is to recognize whether there is a valid SGML application. Some projects that could warrant use of SGML include multiple input sources, output formats and devices, interchange requirements (portability), content identification volume requirements, and multipurpose (multiproduct) databases.

Establish goals for SGML implementation

The second step is to set appropriate goals for the application that will drive an SGML analysis. Goals are the basis for development of the model (DTD) and decide how it is written. Setting goals directly affect publication system design, and might not be attainable with the institutions' current computer system design. A system upgrade may be necessary. Institutions commit to SGML applications usually

because of the need to either interchange, publish, or manage data. Generally SGML is a management decision, and often not the choice of the editorial or publication staff.

Production staffs are primarily concerned with getting the material out on time and may not understand or accept the nature of the new SGML markup system. The previous method may have been working, and production personnel do not understand why they have to go to a new system.

Production may also feel threatened by the data processing nature of SGML. This is because SGML differs from other markup languages in that it has a DTD, a specific set of rules, showing the structure of the documents. This restricts production personnel to placing tags in particular areas and a hierarchical order for compliance. Production may not understand and feel threatened by the way SGML turns documents into a format suitable for electronic display.

SGML can also make the work of production personnel more difficult because of the highly structured, seemingly inflexible process for creating and converting electronic documents. Thus it is critical to see the bigger picture, of identifying the data content, not just getting the document edited and finished in the production stage.

One must review management goals during the initial phase of developing an SGML application, then set both long-term and short-term goals for the SGML application. SGML application goals are outlined in a requirement's document that requires

management approval. Upon approval, the requirements' document is a useful referral and evaluation tool during development of the SGML application. It is important not to stray from the goals outlined in the requirements' document. If goals do change, document the change and retain management approval before continuing development of the SGML application.

In summary, it is important to stay focused on the goals of the SGML application. After management approval, refer to the requirements' document frequently to ensure compatibility during development of the SGML application. Be flexible during development of the SGML application should some goals appear unobtainable. However, developers should never change the goals of an SGML application. Ensure the group of people reach a consensus, particularly management, before going on with changes. During the life of the application, ensure there is adequate personnel familiar with SGML and its terminology. One must continually update and parse the DTD and document instances during the evolution of the project. Finally, evaluate the project by comparing the result against the requirements' document.

Form a working group

The third step is to form a working group to develop the SGML model. The SGML model (DTD) can redefine user roles and their respective turf in relation to others. When forming a

working group one must be prepared to deal with strong egos, opposing points of view, and outright hostility because it will affect their jobs. A good model (DTD) is correct in syntax and semantics, meets the needs of and is acceptable to all user groups, and fosters a sense of ownership among all the groups that will be affected.

Document analysis

The document analysis phase (fourth step) begins by defining the required specifications of the document. An example might be an in-house letter with company specifications that require a front matter, body, and closing matter as its content. Company standards could require the date, addressee, and company title as the front matter. Body matter can include the information sent to the addressee, while the closing provides the author name, company title, enclosures, etc.

After defining document specifications, the working group collects a representative sample of documents for analysis. At the initial stages, the working group should not look at the content and structure of a document and attempt to convert to SGML syntax. They should develop a tree structure to allow non-SGML personnel greater participation and reduce their initial confusion with unfamiliar terms. The working group develops the tree structure by identifying the data elements in the set of documents. Abbreviated names are assigned to

each element (generic identifiers) for eventual insertion into the model.

The tree structure begins with the major structural elements of the document, increasing in complexity as the document is broken into more complex pieces under each major structural element. Document structure is continually broken down until the actual textual information starts entering the tree structure. The working group decides how far to break down the structure during document analysis. For example, if one wishes to search on a database of similar last and first names, the middle initial or name would be vitally important to find the correct one. Tree structure complexity depends on the goals of the SGML application.

In summary, the working group should:

1) Ensure data is tagged sufficiently for efficient database storage and retrieval, and on-screen display.
2) Ensure data is tagged sufficiently to support printed or hardcopy delivery.
3) Ensure that the tagging is appropriate for either unassisted, assisted, or automated tag entry.

Create the Model (DTD)

The fifth step involves the development of a vocabulary representation (DTD) of the tree structure developed in the document analysis phase. The DTD (model) must conform to ISO

8879-1986 Standards, and begin with a DOCTYPE declaration, which specifies the type of document (instance) that follows the DTD (e.g., book, journal, and article). DTDs must be kept as readable as possible to enable review by non-SGML personnel.

Determine the SGML Declaration

The sixth step involves the writing of the SGML Declaration. An SGML application uses rules defined in the SGML Declaration such as the syntax rules, delimiters, and name length for generic identifiers. An SGML Declaration specifies the following:

1) The SGML features in a document.
2) The reference concrete syntax from the ISO 8879-1986 standard to use when tagging documents.
3) Any modifications other than those in the ISO 8879-1986 standard.
4) The character set that is allowable in a valid SGML document.

The ISO 8879-1986 SGML Declaration is an alternative to custom declarations.

SGML Conformance

The seventh step involves SGML Conformance, which includes areas such as conforming document, application, and

system (i.e., from parser to the final product). A conforming SGML application (system) consists of:

1) Conforming documents and relevant documentation.
2) Allowable application conventions.
3) Conformation to the system declaration.
4) Support for reference concrete syntax and reference capacity set.
5) Document parsing facilities for all applications.
6) No enforcement of application conventions as though part of the ISO standard.

SGML Validation

The eighth step requires an SGML parser, DTD, and SGML instance. An SGML parser is the system validator, and an SGML Document Type Definition (DTD) is the model created for the system application. The SGML document is the instance of the SGML application developed.

A parser reads the SGML Declaration, decides if it is syntactically correct, and learns the rules of the SGML application. Then the parser reads in the DTD and decides if it is syntactically correct based on the SGML Declaration. If the DTD is syntactically correct (validated), the parser retrieves the document instance (tagged text file), and compares the syntax and delimiters with those in the SGML Declaration. In the final step the parser verifies that the markup in the instance conforms to the DTD model. An SGML

parser must know the rules in ISO 8879-1986, use and support reference or concrete syntax, parse any DTD by itself, and parse any instance against a DTD. The parser must also find and report markup errors, and not report nonexisting errors.

Document The System That Was Created/Train It/Maintain It

The ninth step includes a document requirements phase, which must be included with any validation process. These documentation requirements must appear in front of all printed publications, for on-screen computer display, and in all training and promotional literature.

System Declaration

A system declaration describes the SGML system. Its form is very similar to the SGML Declaration, which defines such areas as the document features, syntax, variable name length, and maximum quantities. The system declaration gives information on what the system can support. It must meet the same syntax requirements as the SGML Declaration.

APPENDIX C
SELECTED PUBLICATIONS

The number of FCES publications chosen as a representative sample set was fifty. The documents were selected from volumes one and two on FAIRS DISC8. Only documents created after June 1, 1993 were selected to ensure that most publications were tagged with FAST-WP. Each filename begins with two letters that describe the heading under which the publication can be found. The five digit number following the two letters is the actual document number. The filenames and titles that were randomly selected are below.

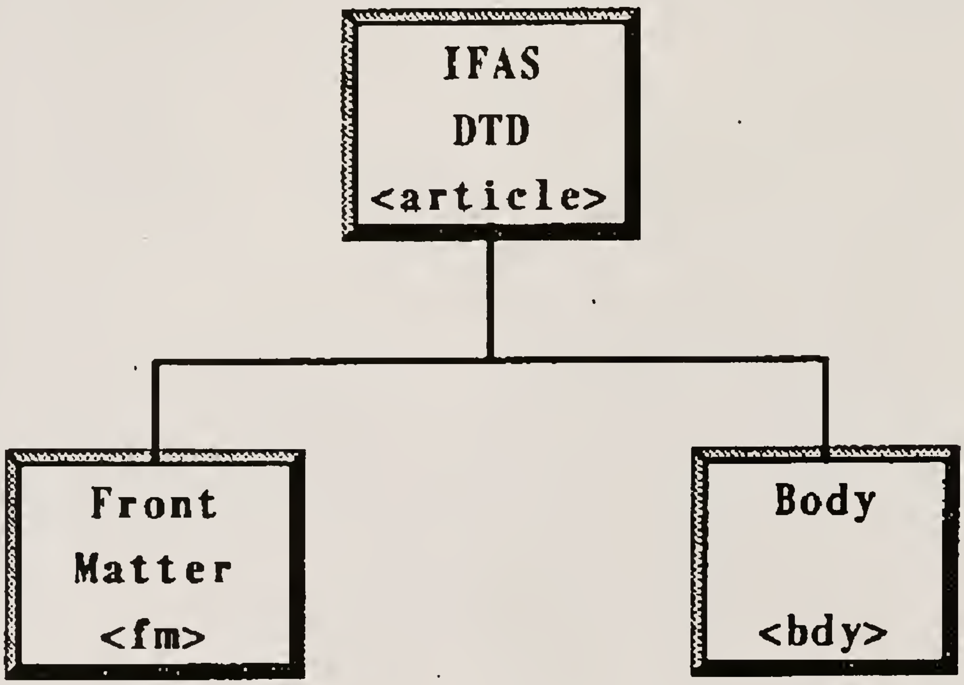
Filename	**Title**
AC01200	Alternative Opportunities for Small Farms: Christmas Tree Production
AC02300	Alternative Opportunities for Small Farms: Pumpkin Production Review
AC02900	Alternative Opportunities for Small Farms: Watermelon Production Review
AE03100	Microirrigation in Florida: Systems, Acreage and Costs
AG00300	Labelled Aquatic Sites for Specific Herbicides
AG01600	Biological Control with Insects: The Hydrilla Stem Weevil
AG01800	Biological Control with Insects: The Waterhyacinth Moth

Filename	Title
AS05800	Limited Certification of Governmental and Private Applicators
AS07700	Florida Pesticide Law and Rules (Florida Statutes -- Chapter 487): Restrictions on the Use of Methyl Bromide
AS09900	Agricultural Safety and Health Educational Materials Directory
AS11200	Farm Respiratory Hazards
DS12100	Water Budgets for Florida Dairy Farms
HE08300	Cradle Crier: Your Child's Development During Month Nine
HE14500	Ayude a Un Nio Despus de Un Huracn: Lo Que Debe Saber
HE15400	Making Financial Plans Together
HE15900	Banking Your Dollars
HE16400	Health Insurance
HE18400	Selecting, Preparing, and Canning: Country Western Ketchup
HE18600	Selecting, Preparing, and Canning: Chile Salsa (Hot Tomato-Pepper Sauce)
HE18700	Desarrollo del Nio: Primer Mes
HE21400	Home Canning: Canning Fruit-Based Baby Foods
HE22800	Selecting, Preparing, and Canning: Figs
HE23400	Selecting, Preparing, and Canning: Nectarines-Halved or Sliced
HE29100	Pickled or Non-Fermented Foods: Pickled Cauliflower or Brussel Sprouts
HE29700	Pickled or Non-Fermented Foods: Pickled Bell Peppers
HE29800	Pickled or Non-Fermented Foods: Pickled Hot Peppers
HE30700	Pickled or Non-Fermented Foods: Quick Sweet Pickles

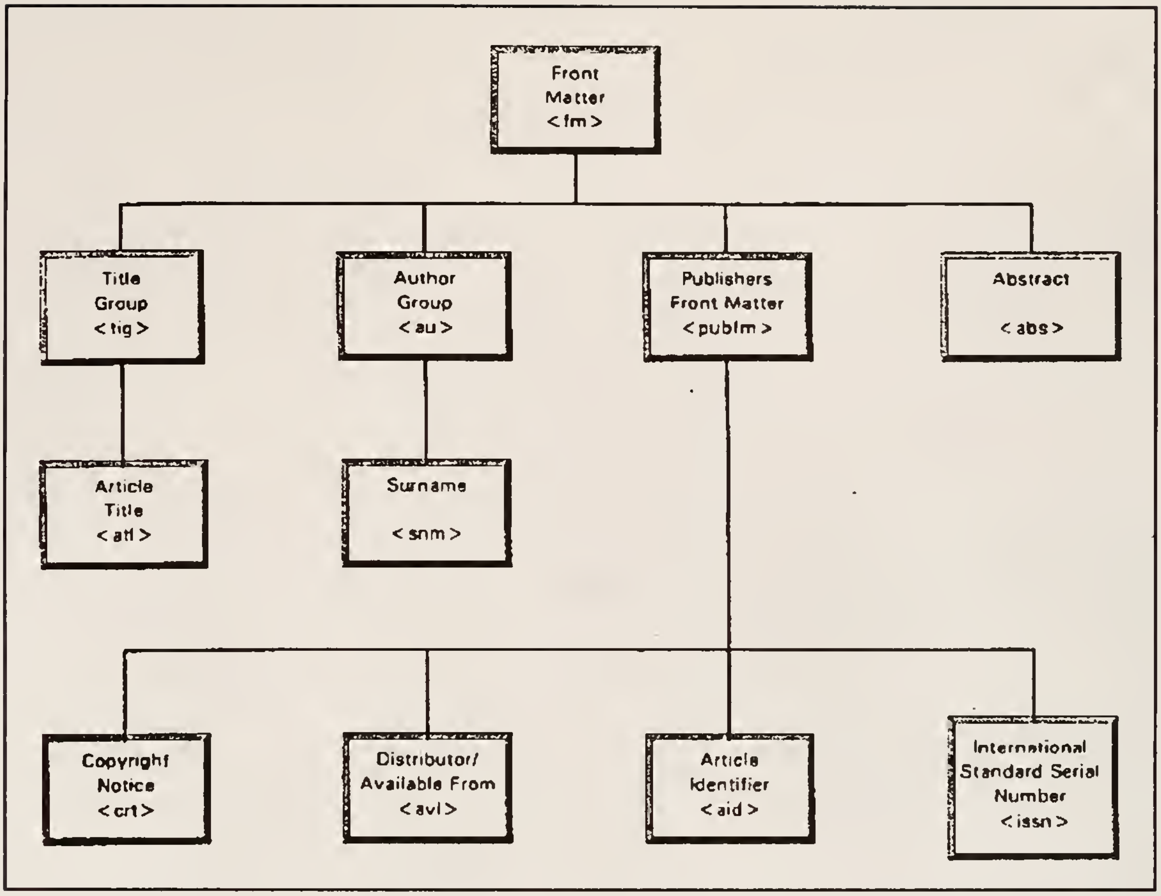
Filename	Title
HE31200	Pickled or Non-Fermented Foods: Pickled Bread-And-Butter Zucchini
HE31600	Jam with Added Pectin: Pear-Apple Jam
HE31900	Jelly with Added Pectin: Grape-Plum Jelly
IG03900	Insect Control in Romaine
IG04200	Insect Control in Sweet Corn
IG04500	Insect Control in Turnips
OA03100	Ventilation: OSHA Standard 1910.94
OA04400	Fire Detection Systems: OSHA Standard 1910.164
PD00100	Phases Of Data Analysis
PD01600	Selecting a Data Collection Technique
SA01000	Safety Signs: ASAE Standard S441
SA02400	Color Coding for Hand Controls: ASAE
SS05700	Okeechobee County: Soil Ratings for Selecting Pesticides
SS08100	Marion Area County: Soil Ratings for Selecting Pesticides
SS09000	St. Lucie County Area: Soil Ratings for Selecting Pesticides
SS10500	Wakulla County: Soil Ratings for Selecting Pesticides
SS10900	Retention of Water Basics of Soil-Water Relationships - Part II
UW00300	Alligators and Crocodiles
UW02700	Owls
UW03600	Snakes
UW03700	Removing Snakes from Dwellings
UW03900	Flying Squirrels
UW04500	Marine Toads

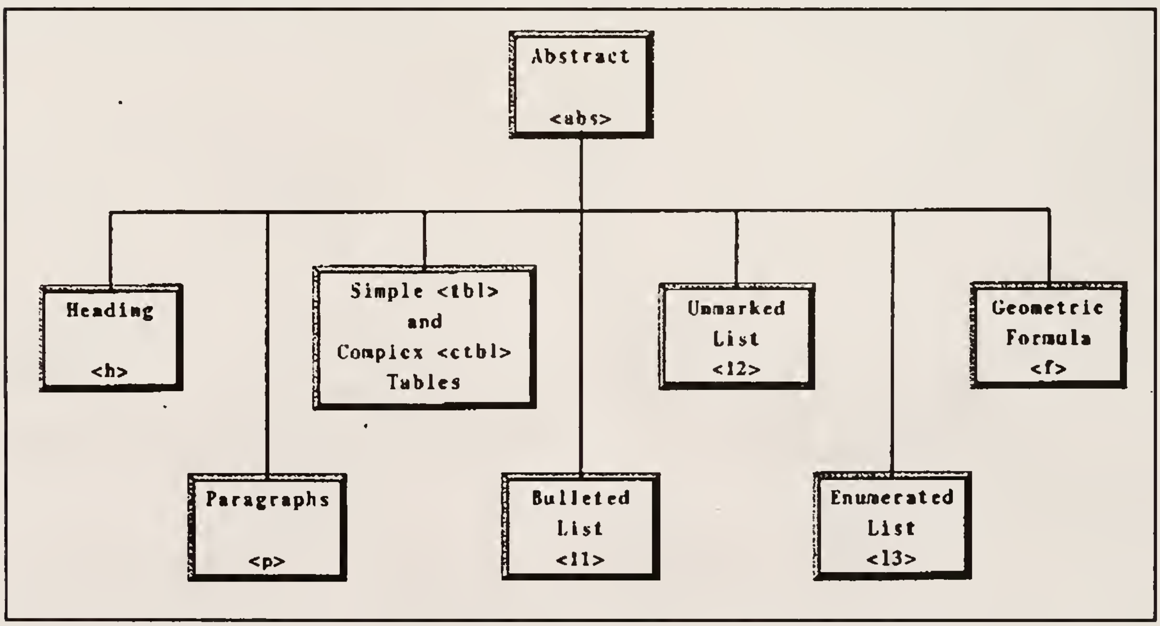
APPENDIX D
TREE STRUCTURE

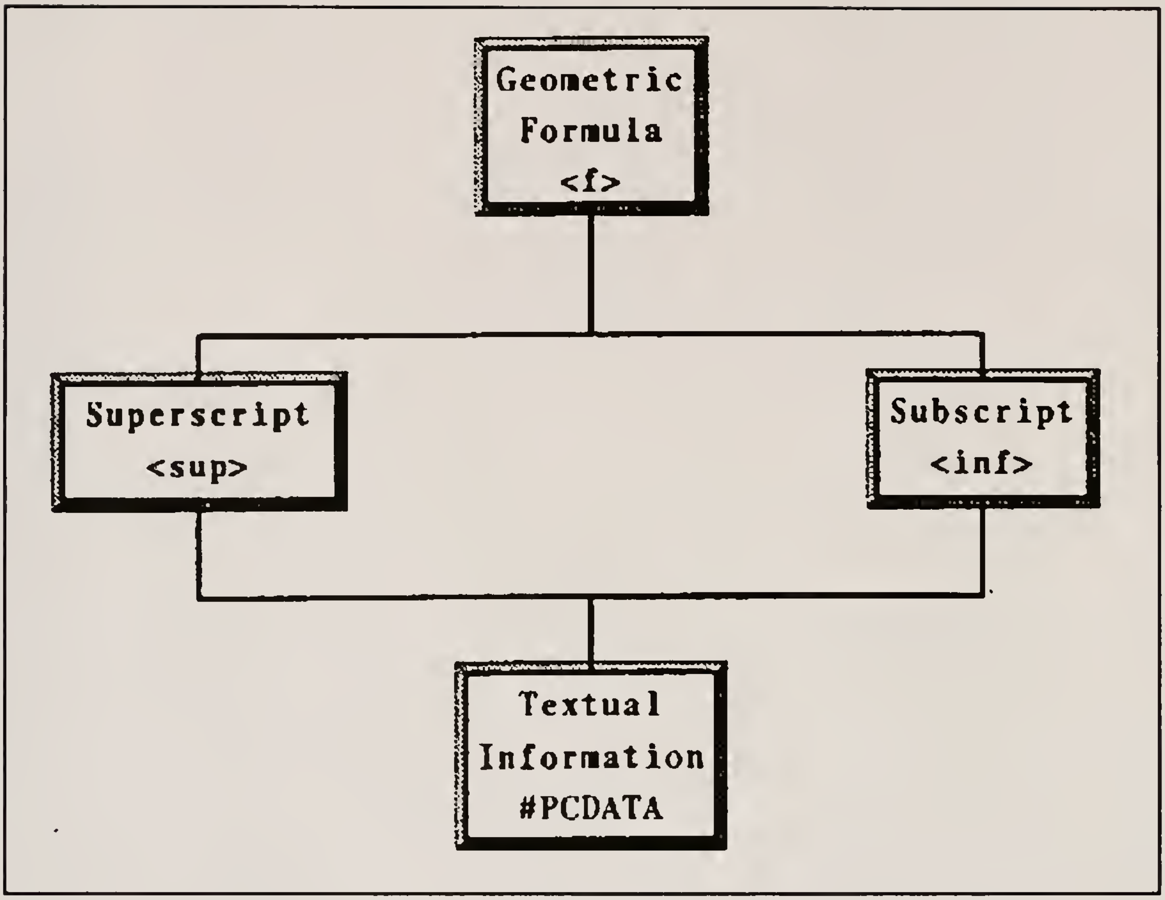
This appendix provides tree diagrams describing the structural properties of elements in the sample set of FCES publications. The tree diagrams were developed from document analysis, the AAP Article model and some unique elements.



Inclusions: Figures <fig>, Footnotes <fn>, and Hyperlinks <hyp>.


Front Matter <fm>
Title Group <tig>
Author Group <au>
Publishers Front Matter <publm>
Abstract <abs>
Article Title <atl>
Surname <snm>
Copyright Notice <crt>
Distributor/ Available From <avl>
Article Identifier <aid>
International Standard Serial Number <issn>


Abstract <abs>
Heading <h>
Simple <tbl> and Complex <ctbl> Tables
Unmarked List <l2>
Geometric Formula <f>
Paragraphs <p>
Bulleted List <l1>
Enumerated List <l3>


Geometric
Formula
<f>
Superscript
<sup>
Subscript
<inf>
Textual
Information
#PCDATA

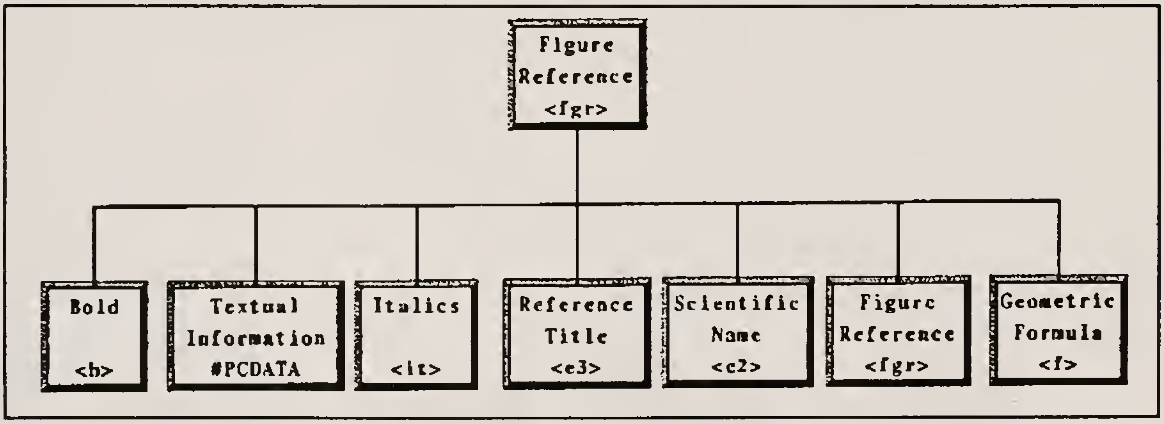
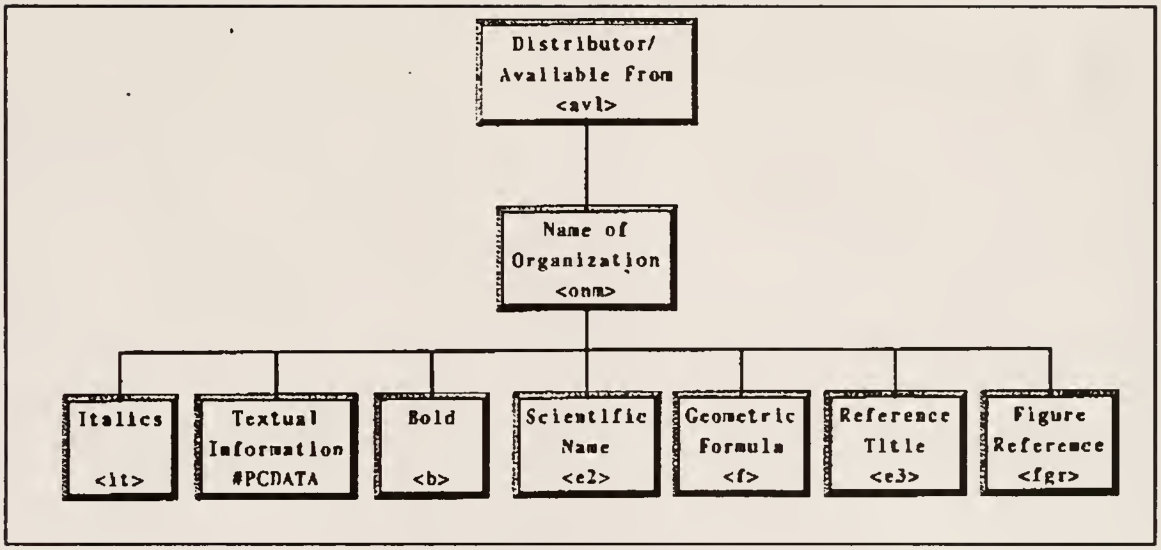
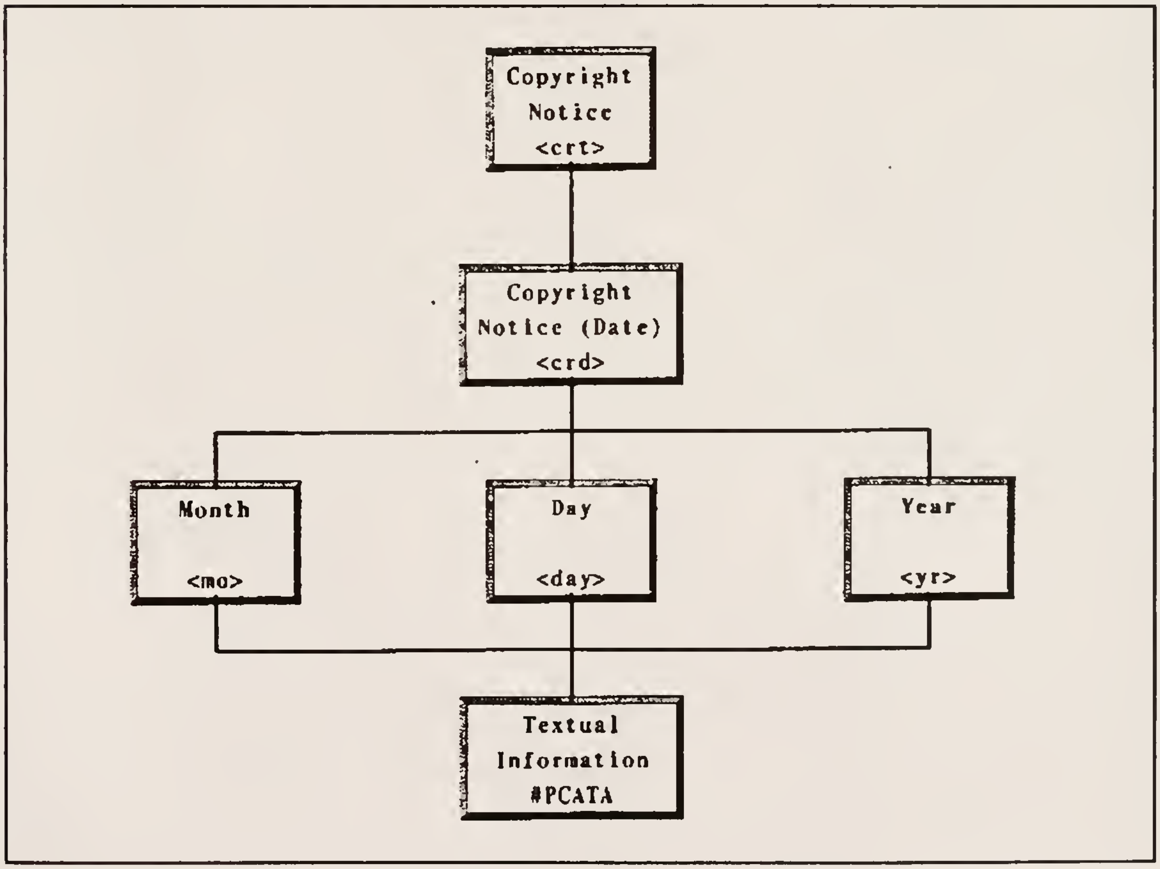
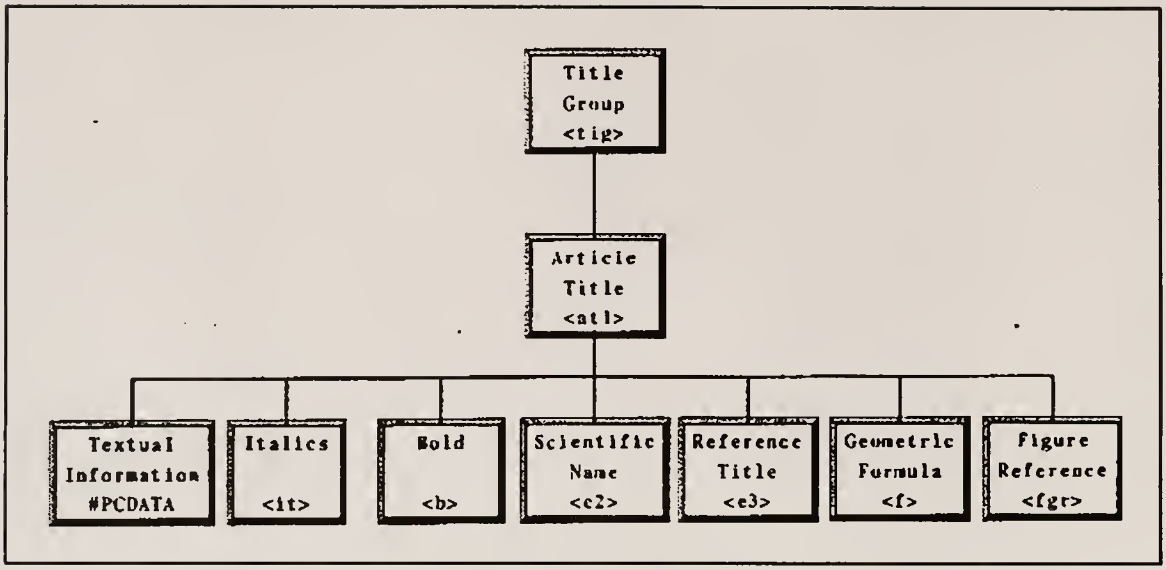
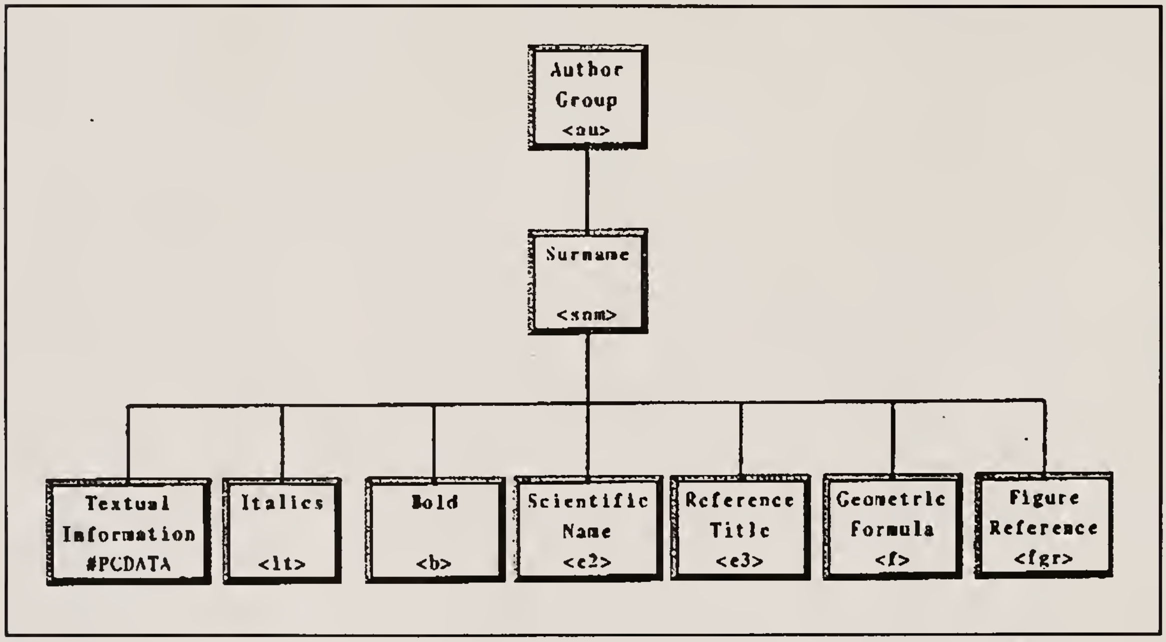


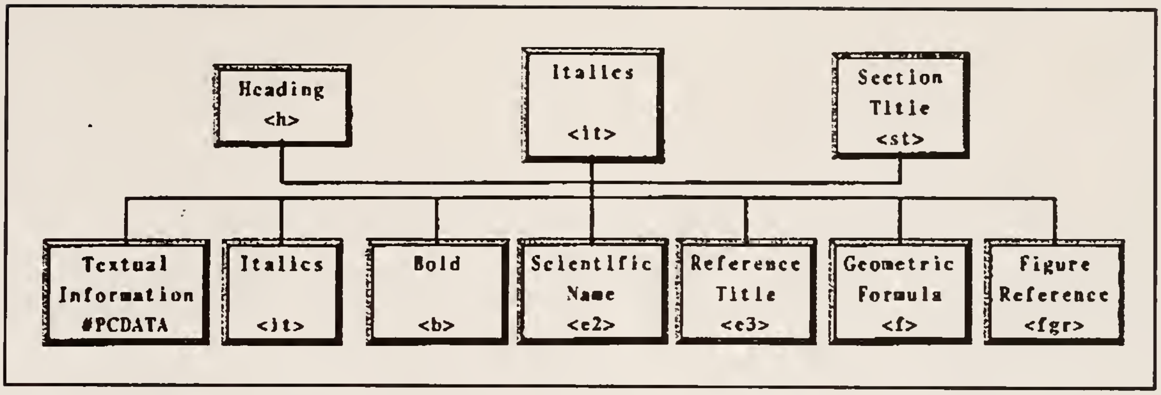
Figure
Reference
<fgr>
Bold
<b>
Textual
Information
#PCDATA
Italics
<it>
Reference
Title
<c3>
Scientific
Name
<c2>
Figure
Reference
<fgr>
Geometric
Formula
<f>

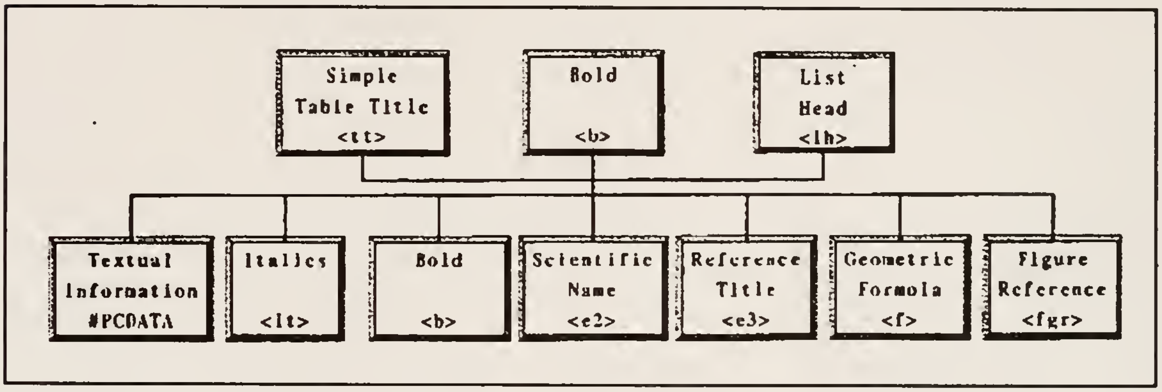

Distributor/
Available From
<avl>
Name of
Organization
<onm>
Italics
<it>
Textual
Information
#PCDATA
Bold
<b>
Scientific
Name
<e2>
Geometric
Formula
<f>
Reference
Title
<e3>
Figure
Reference
<fgr>

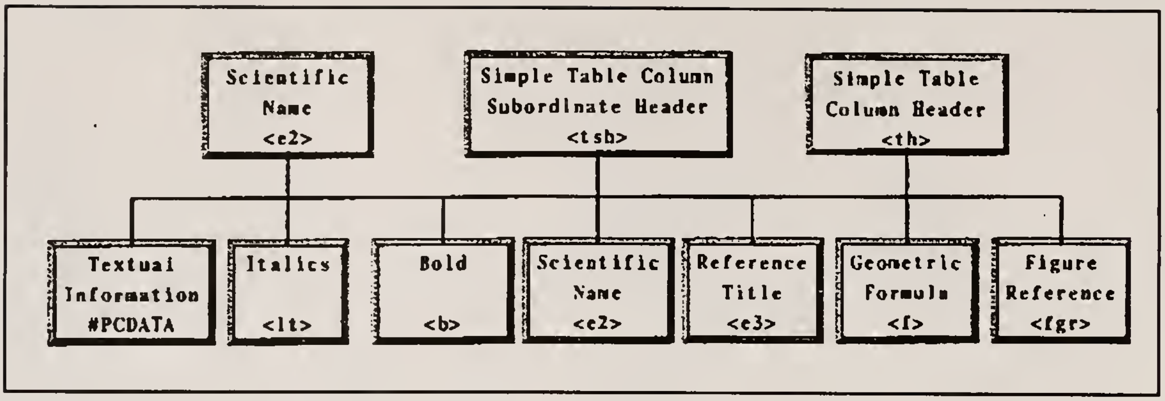

Copyright
Notice
<crt>
Copyright
Notice (Date)
<crd>
Month
<mo>
Day
<day>
Year
<yr>
Textual
Information
#PCATA

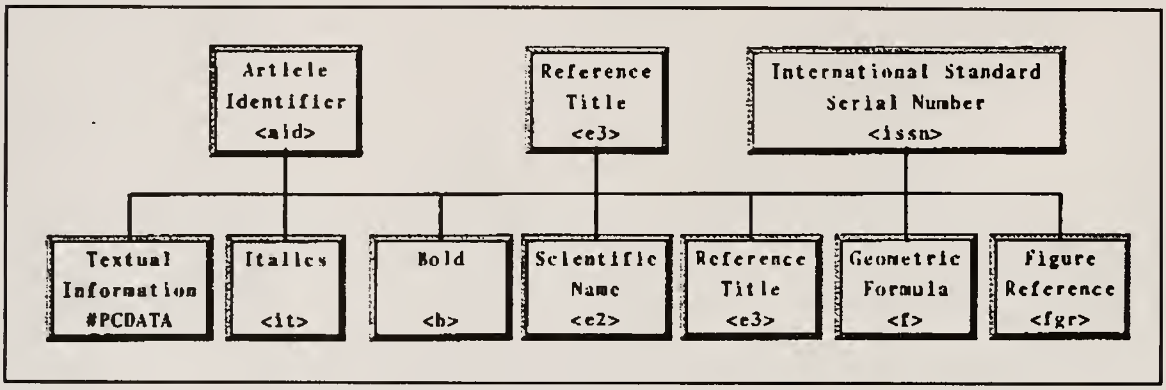

Title Group <tig>
Article Title <atl>
Textual Information #PCDATA
Italics <it>
Bold <b>
Scientific Name <e2>
Reference Title <e3>
Geometric Formula <f>
Figure Reference <fgr>

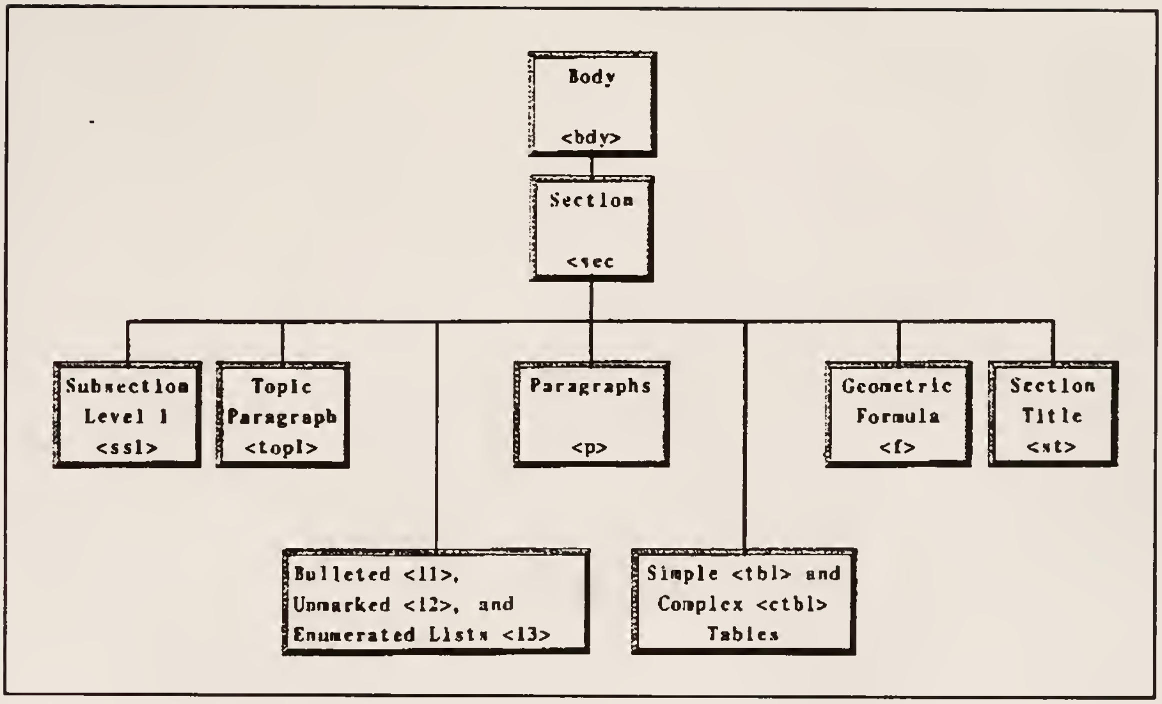

Author Group <au>
Surname <snm>
Textual Information #PCDATA
Italics <it>
Bold <b>
Scientific Name <e2>
Reference Title <e3>
Geometric Formula <f>
Figure Reference <fgr>

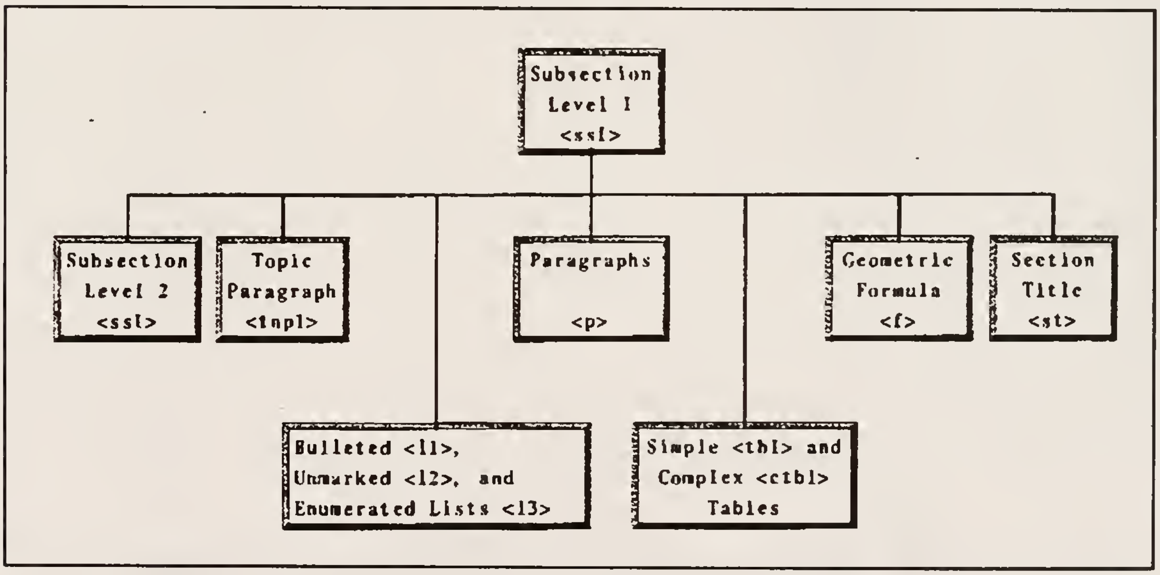

Heading <h>
Italics <it>
Section Title <st>
Textual Information #PCDATA
Italics <it>
Bold <b>
Scientific Name <e2>
Reference Title <e3>
Geometric Formula <f>
Figure Reference <fgr>

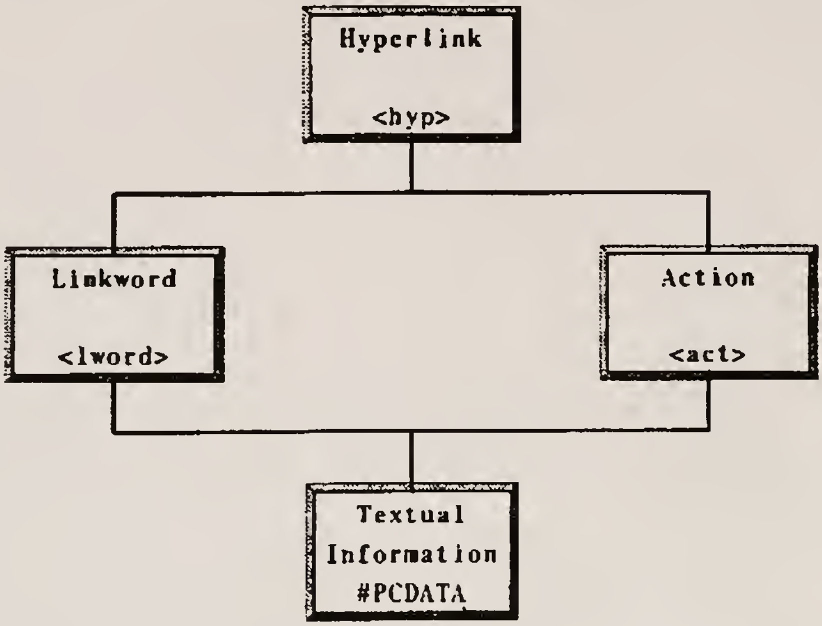

Simple Table Title <tt>
Bold <b>
List Head <lh>
Textual Information #PCDATA
Italics <it>
Bold <b>
Scientific Name <e2>
Reference Title <e3>
Geometric Formula <f>
Figure Reference <fgr>


Scientific Name <e2>
Simple Table Column Subordinate Header <tsb>
Simple Table Column Header <th>
Textual Information #PCDATA
Italics <it>
Bold <b>
Scientific Name <e2>
Reference Title <e3>
Geometric Formula <f>
Figure Reference <fgr>

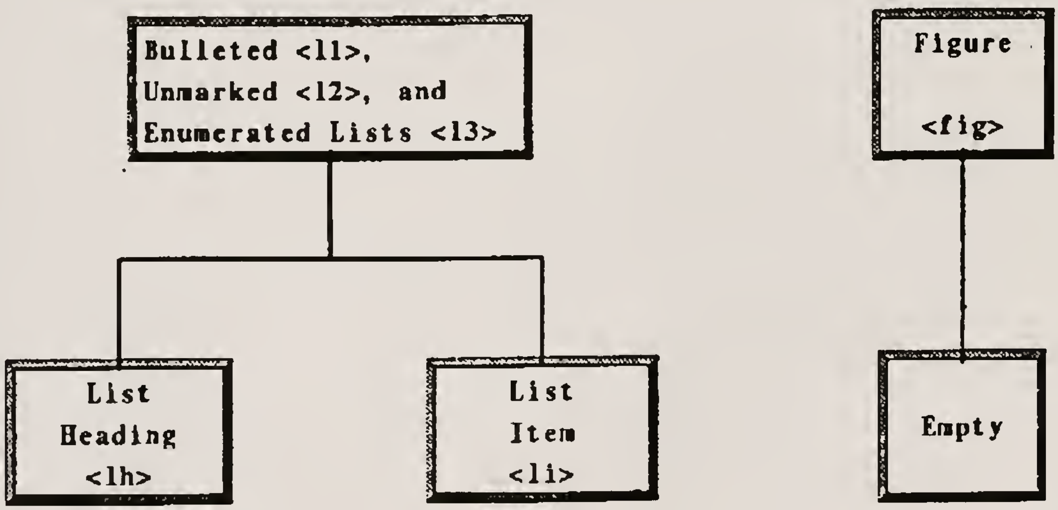

Article Identifier <aid>
Reference Title <e3>
International Standard Serial Number <issn>
Textual Information #PCDATA
Italics <it>
Bold <b>
Scientific Name <e2>
Reference Title <e3>
Geometric Formula <f>
Figure Reference <fgr>

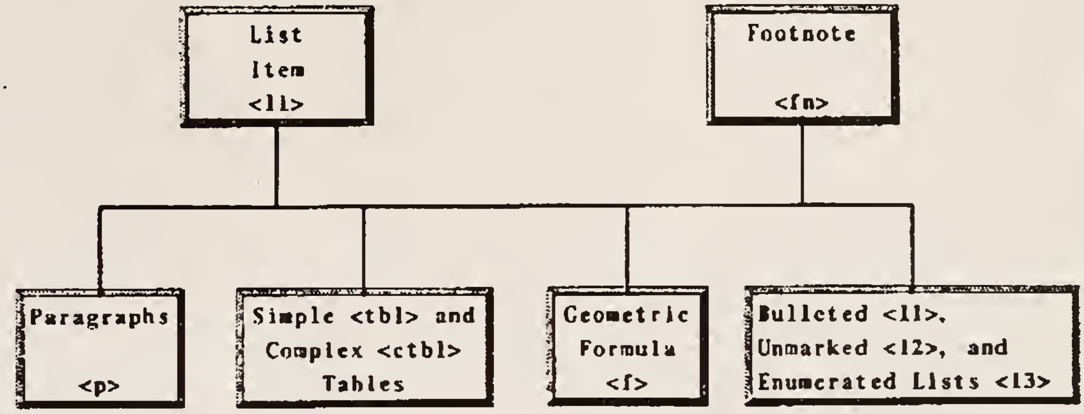

Body
<bdy>
Section
<sec
Subsection
Level 1
<ss1>
Topic
Paragraph
<topl>
Paragraphs
<p>
Geometric
Formula
<f>
Section
Title
<st>
Bulleted <l1>,
Unmarked <l2>, and
Enumerated Lists <l3>
Simple <tbl> and
Complex <ctbl>
Tables


Subsection
Level 1
<ss1>
Subsection
Level 2
<ss1>
Topic
Paragraph
<tnpl>
Paragraphs
<p>
Geometric
Formula
<f>
Section
Title
<st>
Bulleted <l1>,
Unmarked <l2>, and
Enumerated Lists <l3>
Simple <tbl> and
Complex <ctbl>
Tables

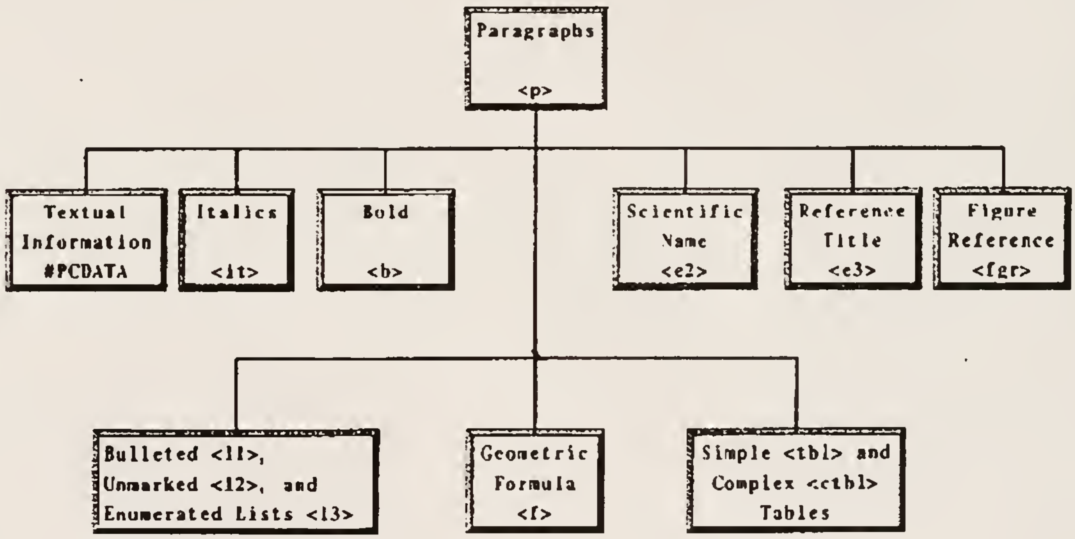


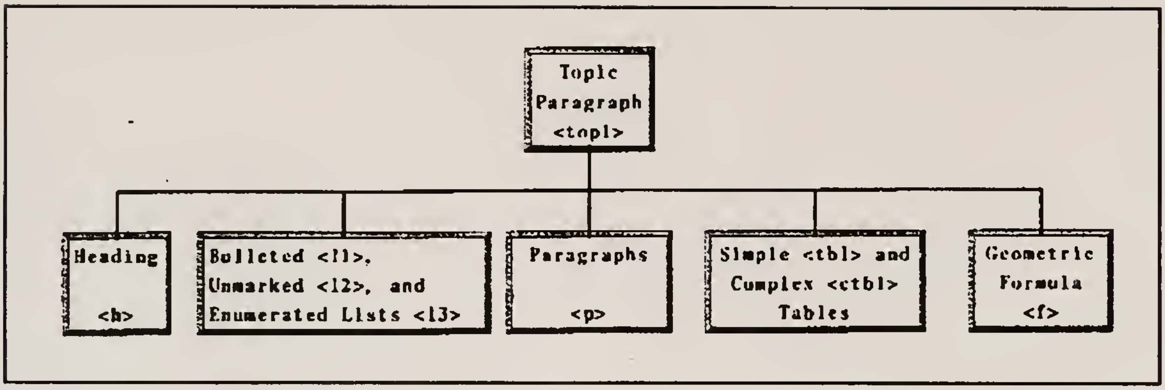
Exclusions: Figures <fig>, Footnotes <fn>, and Hyperlinks <hyp>.

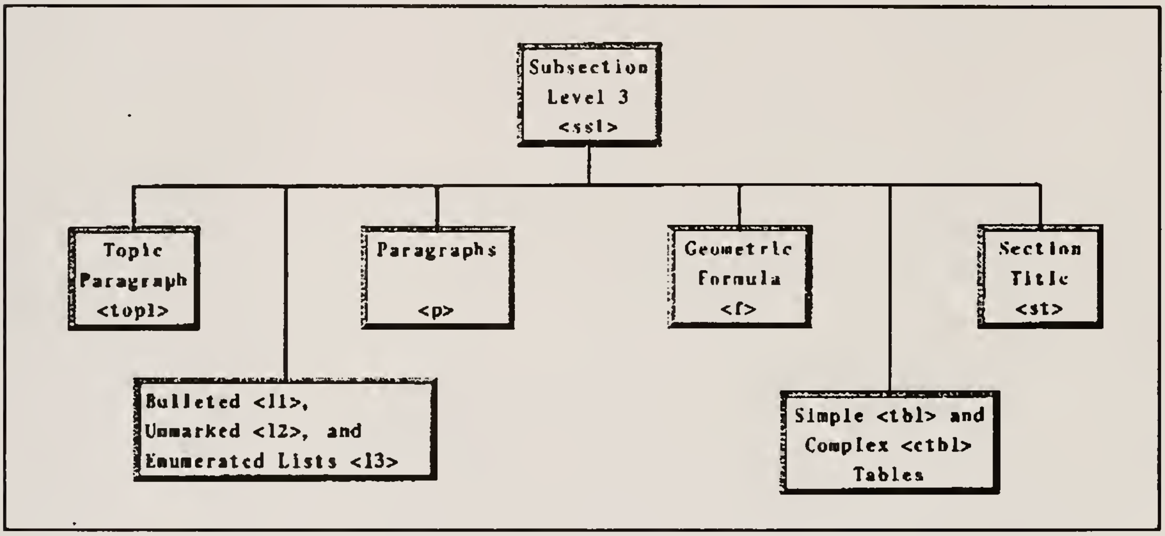


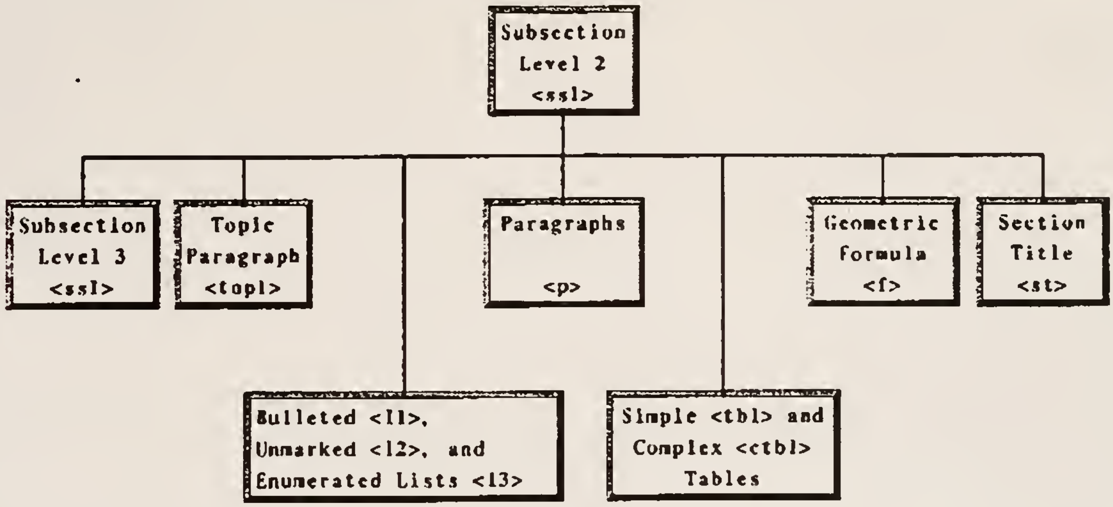


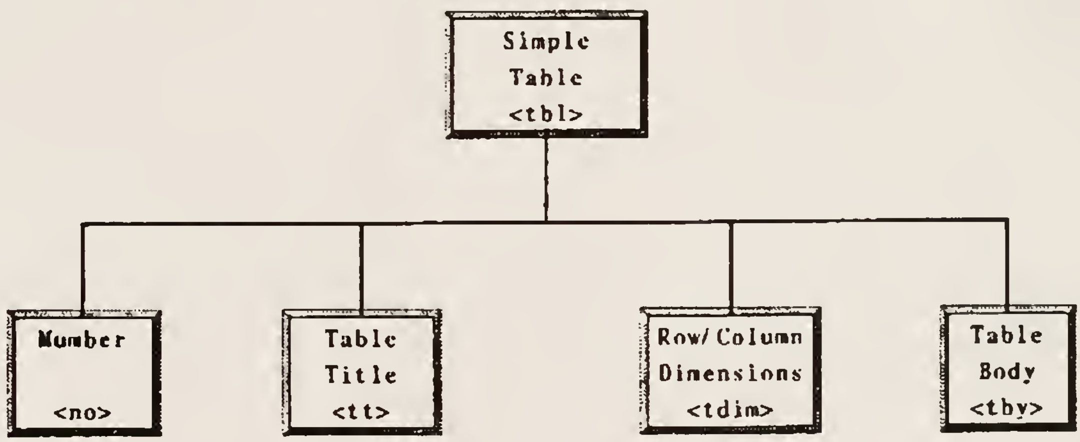
Exclusions in Footnotes <fn>: Figures <fig> and Footnotes <fn>.



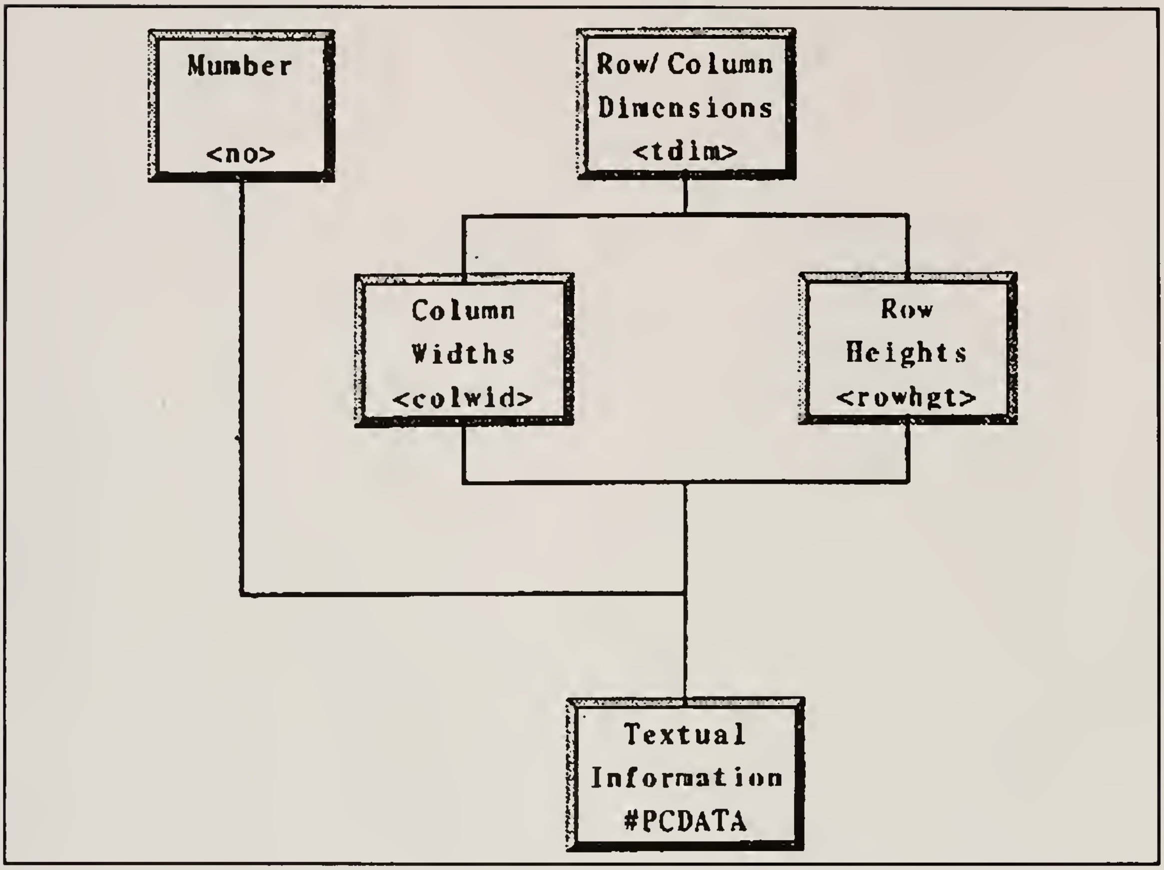

Topic Paragraph <topl>
Heading <h>
Bulleted <l1>, Unmarked <l2>, and Enumerated Lists <l3>
Paragraphs <p>
Simple <tbl> and Complex <ctbl> Tables
Geometric Formula <f>


Subsection Level 3 <ss1>
Topic Paragraph <topl>
Paragraphs <p>
Geometric Formula <f>
Section Title <st>
Bulleted <l1>, Unmarked <l2>, and Enumerated Lists <l3>
Simple <tbl> and Complex <ctbl> Tables





Exclusions: Figures <fig>, Footnotes <fn>, Simple <tbl> Complex <ctbl> Tables.


Number
<no>
Row/Column
Dimensions
<tdim>
Column
Widths
<colwid>
Row
Heights
<rowhgt>
Textual
Information
#PCDATA

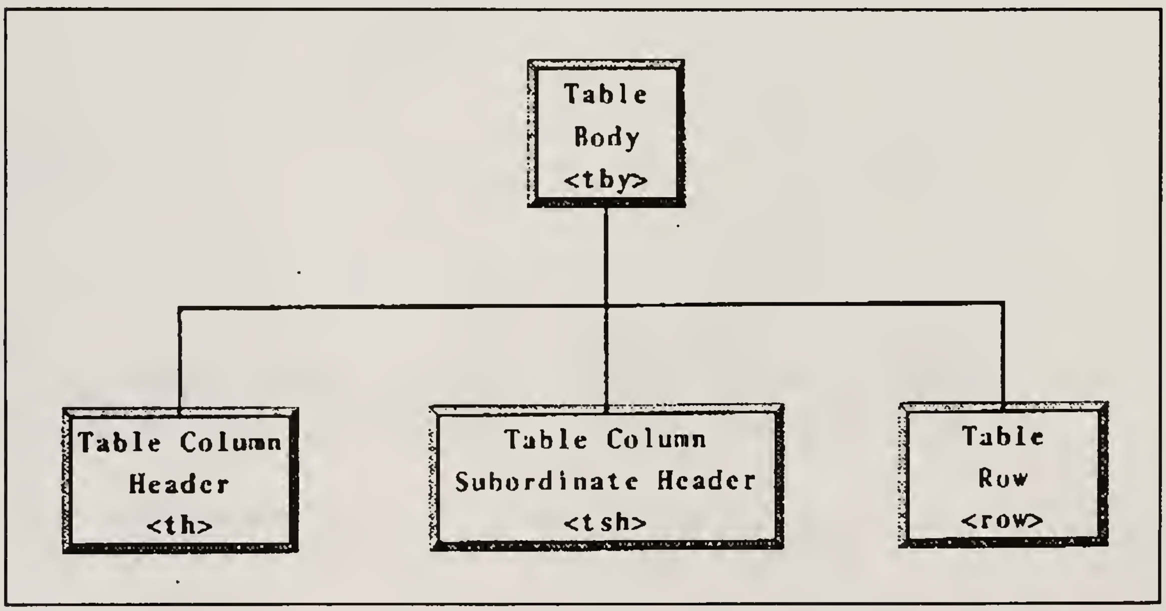
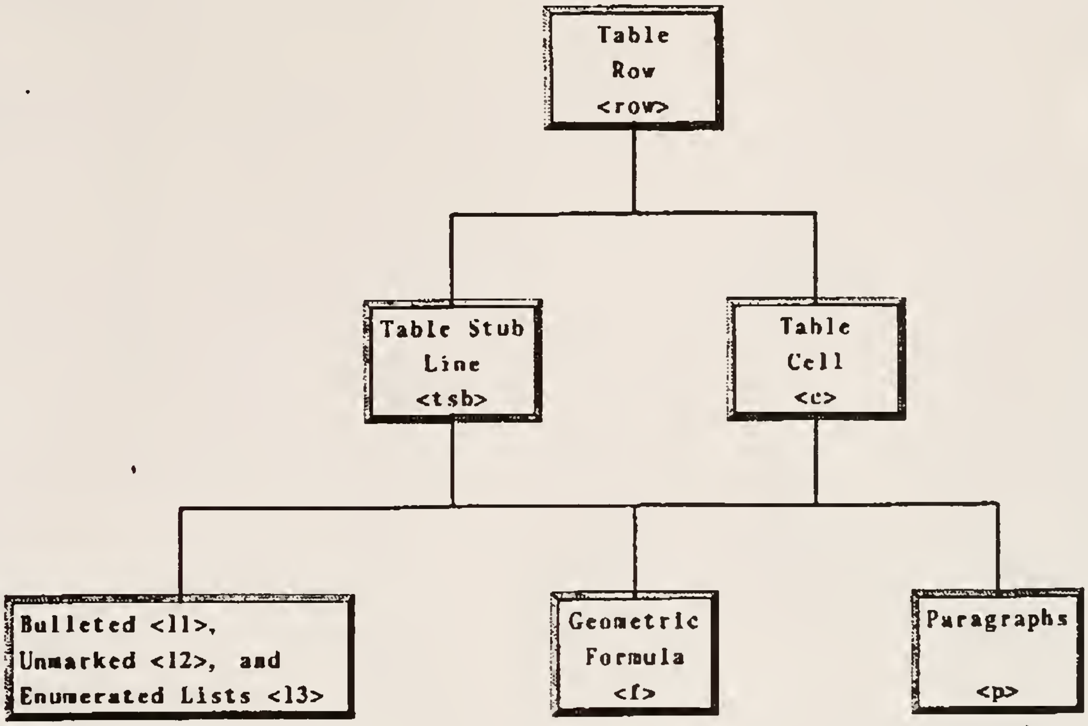
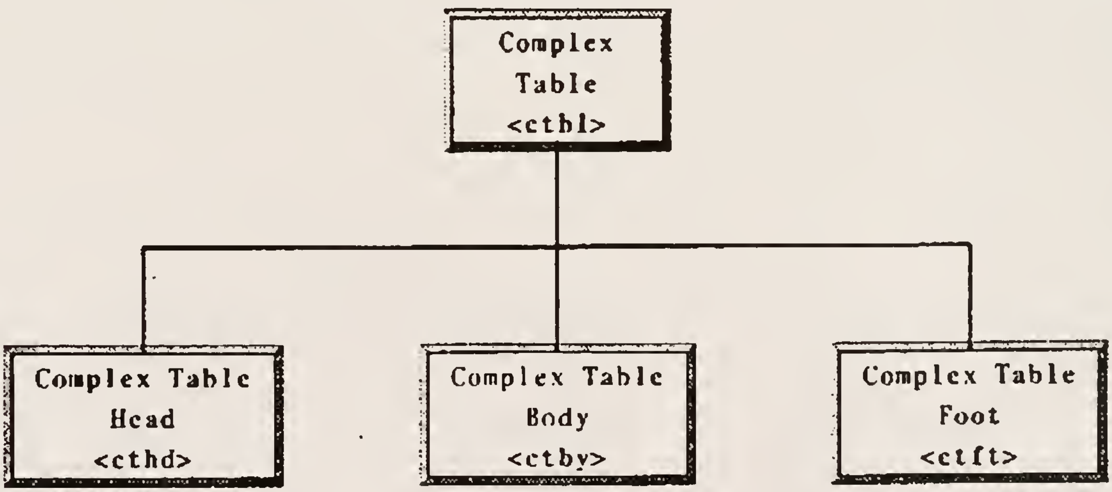
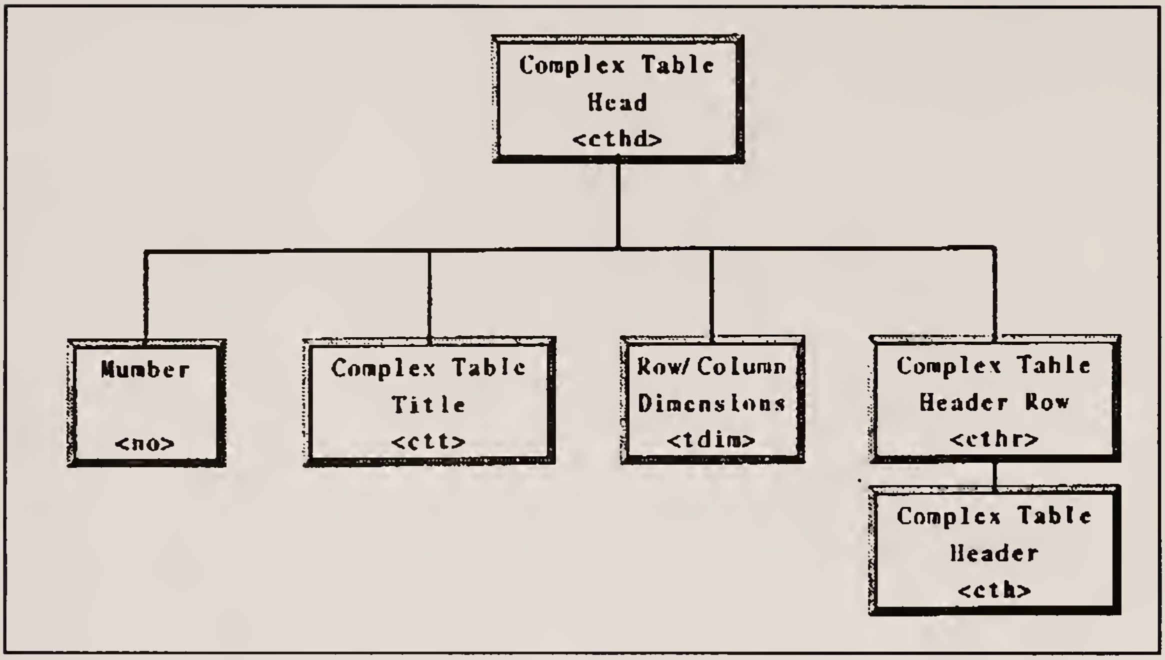


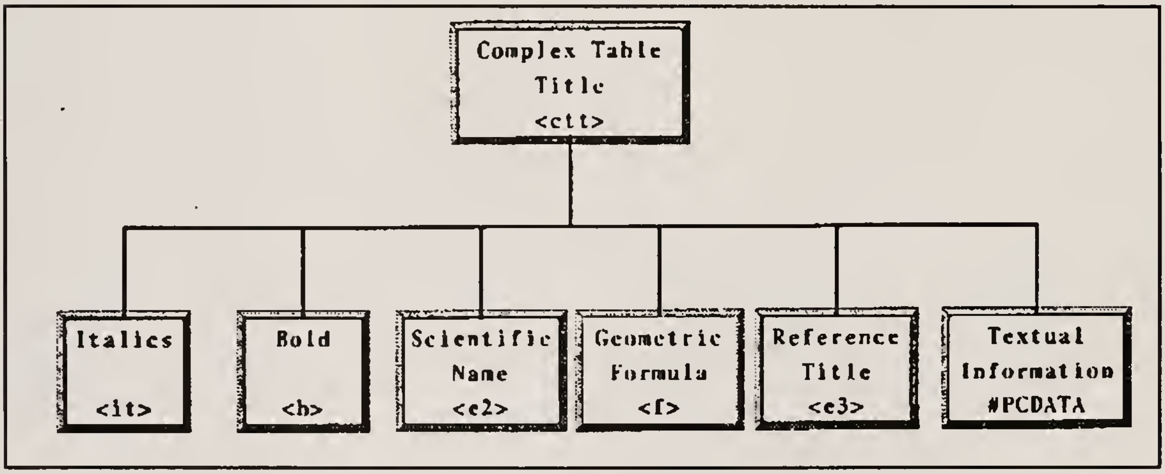
Table
Body
<tby>
Table Column
Header
<th>
Table Column
Subordinate Header
<tsh>
Table
Row
<row>

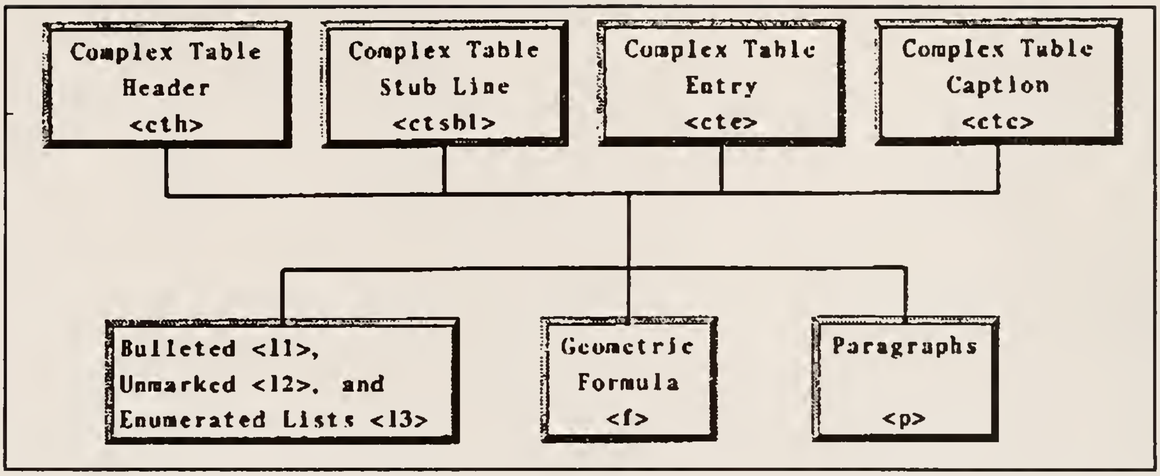


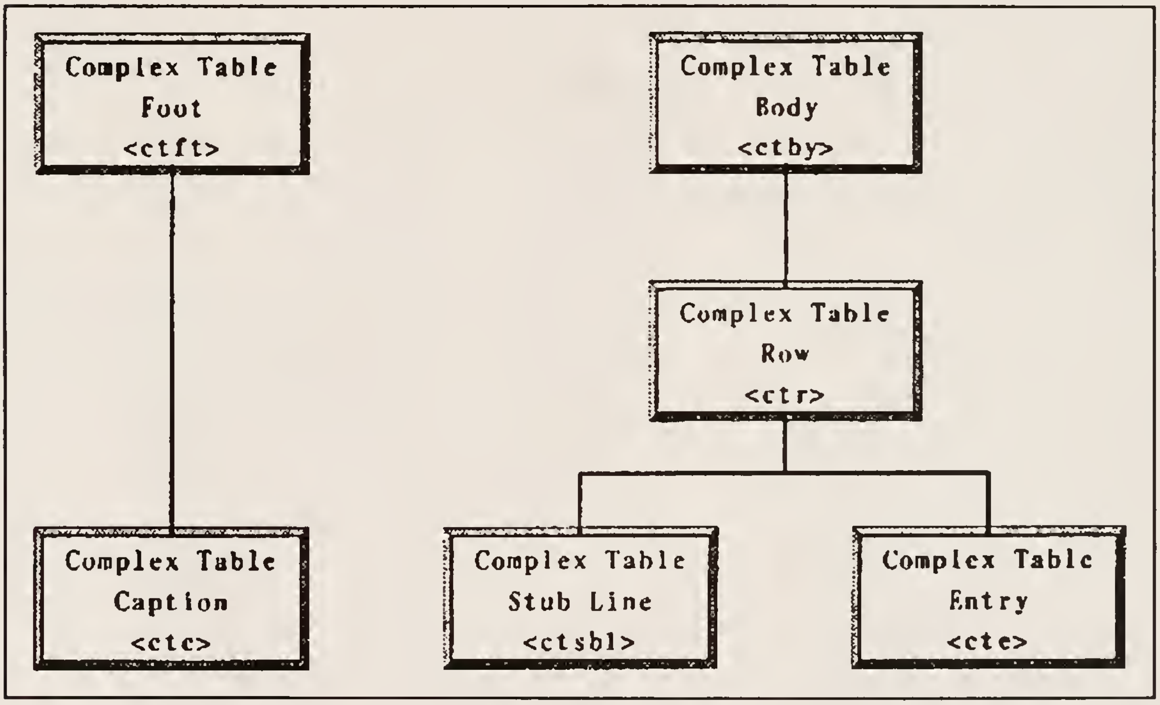


Exclusions: Figures <fig>, Footnotes <fn>, Simple <tbl> Complex <ctbl> Tables.


Complex Table Head <cthd>
Number <no>
Complex Table Title <ctt>
Row/Column Dimensions <tdim>
Complex Table Header Row <cthr>
Complex Table Header <cth>


Complex Table Title <ctt>
Italics <it>
Bold <b>
Scientific Name <e2>
Geometric Formula <f>
Reference Title <e3>
Textual Information #PCDATA


Complex Table Header <cth>
Complex Table Stub Line <ctsbl>
Complex Table Entry <cte>
Complex Table Caption <ctc>
Bulleted <l1>, Unmarked <l2>, and Enumerated Lists <l3>
Geometric Formula <f>
Paragraphs <p>


Complex Table Foot <ctft>
Complex Table Body <ctby>
Complex Table Row <ctr>
Complex Table Caption <ctc>
Complex Table Stub Line <ctsbl>
Complex Table Entry <cte>

APPENDIX E
MODEL ELEMENTS, ATTRIBUTES, AND ENTITIES

This appendix provides a comprehensive list of the elements, attributes and entities in the FCES model. Each generic identifier (name) is either an AAP or University of Florida element tag. Descriptions are provided below for each element, attribute and entity.

TAG NAME	COMPREHENSIVE ELEMENT LIST		
	DESCRIPTION	AAP Compliant (y/n)	Notes
<article>	Type of Instance	y	Article DTD is the only one in use.
<fm>	Front Matter	y	Front-end document information through author.
<tig>	Title Group	y	Title Information.
<atl>	Article Title	y	Article Title.
<it>	Italics	y	
<b>	Bold	y	
<e1>	Straight Emphasis	y	Not used. Bold and Italics are emphasis.
<e2>	Scientific Name	y	
<e3>	Reference Title	y	The title of a reference document.
<fgr>	Figure, A Reference To (fgr rid=fg*)	y	Not used.
<f>	Geometric Formula	y	Currently used only to tag superscript and subscript.
<sup>	Superscript	y	

TAG NAME	COMPREHENSIVE ELEMENT LIST		
	DESCRIPTION	AAP Compliant (y/n)	Notes
<inf>	Subscript	y	
<au>	Author (Name Component Parts)	y	
<snm>	Surname	y	Surname is used as tag for entire name.
<pubfm>	Publisher's Front Matter Group	y	Used for copyright.
<crt>	Copyright Notice	y	"
<crd>	Copyright Notice (Date)	y	"
<avl>	Distributor/Available From	y	"
<onm>	Name of Organization	y	"
<aid>	Article Identifier	y	"
<issn>	International Standard Serial Number	y	"
<abs>	Abstract	y	
<mo>	Month	y	
<day>	Day	y	
<yr>	Year	y	

TAG NAME	COMPREHENSIVE ELEMENT LIST		
	DESCRIPTION	AAP Compliant (y/n)	Notes
<h>	Heading	y	Only used with a paragraph head.
<bdy>	Body Matter	y	Main body of publication text after author.
<sec>	Section	y	
<ss1>	Subsection Level 1	y	
<ss2>	Subsection Level 2	y	
<ss3>	Subsection Level 3	y	
<st>	Section Title	y	
<p>	Paragraph	y	Section Heading.
<top1>	Topic Paragraph (Captioned), Type 1	y	Known as a paragraph head.
<l1>	Bulleted List	y	
<l2>	Unmarked List	y	
<l3>	Enumerated List	y	
	List Item	y	
<lh>	List Head	y	

TAG NAME	COMPREHENSIVE ELEMENT LIST		
	DESCRIPTION	AAP Compliant (y/n)	Notes
<fig>	Figure	y	
<fn>	Footnote	y	
<hyp>	Hyperlink	**UF**	
<lword>	Link Word (Hyperlink)	**UF**	
<act>	Action (Hyperlink)	**UF**	
<tbl>	Simple Table	y	
<tt>	Table Title	y	
<tby>	Table Body	y	
<row>	Table Row	y	
<th>	Table Column Header	y	
<tsh>	Table Column Subordinate Header	y	
<tsb>	Table Stub Line	y	
<c>	Cell	y	
<ctbl>	Complex Table	y	
<cthd>	Complex Table Head	y	
<ctby>	Complex Table Body	y	

TAG NAME	COMPREHENSIVE ELEMENT LIST		
	DESCRIPTION	AAP Compliant (y/n)	Notes
<ctbf>	Complex Table Foot	y	
<no>	Number	y	
<ctt>	Complex Table Title	y	
<cthr>	Complex Table Header Row	y	
<tdim>	Row/Column Dimensions for Simple and Complex Table	**UF**	Sets rectangular coordinates of tables.
<colwid>	Column Widths in Inches for Simple and Complex Table	**UF**	
<rowhgt>	Row Widths in Inches for a Simple and Complex Table	**UF**	
<cth>	Complex Table Header	y	
<ctr>	Complex Table Row	y	
<ctsb1>	Complex Table Stub Line, Level 1	y	First column of complex table.
<cte>	Complex Table Entry	y	Data in each cell.
<ctc>	Complex Table Identifier/Caption	y	Can be in or separate from table.

Name of Element to which the list belongs	Attribute Definition List				
	Element Attribute Names	Declared Attribute Value	Explanation	Tag Type	Notes
act	type	text	Hyperlink to a text file	*UF*	
act	type	bitmap	Hyperlink for a slide, etc.	*UF*	
act	type	exeprgm	Hyperlink to Run a Program	*UF*	
act	type	audio	Hyperlink for an Audio Program	*UF*	
act	type	table	Hyperlink to a Table	*UF*	
act	type	datarec	Hyperlink to a data record	*UF*	
act	descr	CDATA	Character Data in Comment Box	*UF*	
cth	align	l	Left Alignment	AAP	
cth	align	c	Center Alignment	AAP	
cth	align	r	Right Alignment	AAP	
cth	align	d	Decimal Alignment	AAP	
cth	valign	t	Top Alignment	AAP	

Name of Element to which the list belongs	Attribute Definition List				
	Element Attribute Names	Declared Attribute Value	Explanation	Tag Type	Notes
cth	valign	m	Middle Alignment	AAP	
cth	valign	b	Bottom Alignment	AAP	
cth	cb	NUMBER	Column Beginning Number When Straddling More Than One Column	AAP	
cth	ce	NUMBER	Column Ending Number When Straddling More Than One Column	AAP	
cth	rb	NUMBER	Row Beginning Number When Straddling More Than One Row	AAP	
cth	re	NUMBER	Row Ending Number When Straddling More Than One Row	AAP	
c	shaded	(y/n)	Determines if cell data are shaded(y) or not(n).	*UF*	An aid for retrieval software.

Name of Element to which the list belongs	Attribute Definition List				
	Element Attribute Names	Declared Attribute Value	Explanation	Tag Type	Notes
c	topline	NAME	Defines whether the top of each cell has either (n)o, (s)ingle, (d)ouble, d(a)shed, d(o)tted, (t)hick, or (e)xtra-thick line.	*UF*	An aid for retrieval software.
c	botline	NAME	Defines whether the bottom of each cell has either (n)o, (s)ingle, (d)ouble, d(a)shed, d(o)tted, (t)hick, or (e)xtra-thick line.	*UF*	An aid for retrieval software.
c	lftline	NAME	Defines whether the left of each cell has either (n)o, (s)ingle, (d)ouble, d(a)shed, d(o)tted, (t)hick, or (e)xtra-thick line.	*UF*	An aid for retrieval software.

Name of Element to which the list belongs	Attribute Definition List				
	Element Attribute Names	Declared Attribute Value	Explanation	Tag Type	Notes
c	rgtline	NAME	Defines whether the right of each cell has either (n)o, (s)ingle, (d)ouble, d(a)shed, d(o)tted, (t)hick, or (e)xtra-thick line.	*UF*	An aid for retrieval software.
ctsb1	align	l	Left Alignment	AAP	
ctsb1	align	c	Center Alignment	AAP	
ctsb1	align	r	Right Alignment	AAP	
ctsb1	align	d	Decimal Alignment	AAP	
ctsb1	valign	t	Top Alignment	AAP	
ctsb1	valign	m	Middle Alignment	AAP	
ctsb1	valign	b	Bottom Alignment	AAP	
cte	align	l	Left Alignment	AAP	
cte	align	c	Center Alignment	AAP	
cte	align	r	Right Alignment	AAP	
cte	align	d	Decimal Alignment	AAP	

Name of Element to which the list belongs	Attribute Definition List				
	Element Attribute Names	Declared Attribute Value	Explanation	Tag Type	Notes
cte	valign	t	Top Alignment	AAP	
cte	valign	m	Middle Alignment	AAP	
cte	valign	b	Bottom Alignment	AAP	
cte	cb	NUMBER	Column Beginning Number When Straddling More Than One Column	AAP	
cte	ce	NUMBER	Column Ending Number When Straddling More Than One Column	AAP	
cte	rb	NUMBER	Row Beginning Number When Straddling More Than One Row	AAP	
cte	re	NUMBER	Row Ending Number When Straddling More Than One Row	AAP	
cte	shaded	(y/n)	Determines if cell data are shaded(y) or not(n).	*UF*	An aid for retrieval software.

Name of Element to which the list belongs	Attribute Definition List				
	Element Attribute Names	Declared Attribute Value	Explanation	Tag Type	Notes
cte	topline	NAME	Defines whether the top of each cell has either (n)o, (s)ingle, (d)ouble, d(a)shed, d(o)tted, (t)hick, or (e)xtra-thick line.	*UF*	An aid for retrieval software.
cte	botline	NAME	Defines whether the bottom of each cell has either (n)o, (s)ingle, (d)ouble, d(a)shed, d(o)tted, (t)hick, or (e)xtra-thick line.	*UF*	An aid for retrieval software.
cte	lftline	NAME	Defines whether the left of each cell has either (n)o, (s)ingle, (d)ouble, d(a)shed, d(o)tted, (t)hick, or (e)xtra-thick line.	*UF*	An aid for retrieval software.

Name of Element to which the list belongs	Attribute Definition List				
	Element Attribute Names	Declared Attribute Value	Explanation	Tag Type	Notes
cte	rgtline	NAME	Defines whether the right of each cell has either (n)o, (s)ingle, (d)ouble, d(a)shed, d(o)tted, (t)hick, or (e)xtra-thick line.	*UF*	An aid for retrieval software.

FAST-WP Symbols with Entity References			
Symbols	Entity Reference	Symbol/Entity Description	Notes
&	&	ampersand	Start of an entity.
<	<	less than sign	Open tag.
]	]	right square bracket	End of a DTD.
á	á	small a with acute accent	Non-ASCII characters from here out on the rows.
Á	Á	Capital A with acute accent	
ä	ä	small a with diaeresis or umlaut mark	
Ä	Ä	Capital A with diaeresis or umlaut mark	
é	é	small e with acute accent	
É	É	Capital E with acute accent	
í	í	small i with acute accent	
Í	Í	Capital I with acute accent	
ó	ó	small o with acute accent	
Ó	Ó	Capital O with acute accent	
ö	ö	small o with diaeresis or umlaut mark	

FAST-WP Symbols with Entity References			
Symbols	Entity Reference	Symbol/Entity Description	Notes
Ö	Ö	Capital O with diaeresis or umlaut mark	
ú	ú	small u with acute accent	
Ú	Ú	Capital U with acute accent	
ü	ü	small u with diaeresis or umlaut mark	
Ü	Ü	Capital U with diaeresis or umlaut mark	
•	•	bullet	
■	▪	solid small square; solid en quad	
♦	♦	diamond suit symbol	
¢	¢	cents sign	
✓	✓	check mark	
©	©	copyright symbol	
–	–	en dash	
—	—	em dash	
°	°	degree sign	

FAST-WP Symbols with Entity References

Symbols	Entity Reference	Symbol/Entity Description	Notes
′	′	single prime or minute (**feet**)	
″	″	double prime or second (**inches**)	
♀	♀	female symbol	
α	&agr;	small alpha, Greek	
β	&bgr;	small beta, Greek	
δ	&dgr;	small delta, Greek	
Δ	&Dgr;	capital Delta, Greek	
μ	&mgr;	small mu, Greek	
σ	&sgr;	small sigma, Greek	
Σ	&Sgr;	capital Sigma, Greek	
♂	♂	male symbol	
−	−	minus sign	
×	×	multiplication sign	
÷	÷	divided by sign	
±	±	plus-or-minus sign	
≤	≤	less-than-or-equal sign	

FAST-WP Symbols with Entity References

Symbols	Entity Reference	Symbol/Entity Description	Notes
≥	≥	greater-than-or-equal sign	
≠	≠	not-equal	
½	½	fraction one-half	
⅓	⅓	fraction one-third	
¼	¼	fraction one-quarter	
⅔	⅔	fraction two-thirds	
¾	¾	fraction three-quarters	
Ñ	Ñ	capital N with tilde	
ñ	ñ	small n with tilde	
“	“	left double quotation mark	
”	”	right double quotation mark	
®	®	registered sign	
™	™	trademark sign	
¿	¿	inverted question mark	
¡	¡	inverted exclamation mark	
«	«	left angle quotation mark	

FAST-WP Symbols with Entity References			
Symbols	Entity Reference	Symbol/Entity Description	Notes
»	»	right angle quotation mark	

APPENDIX F
STRUCTURE OF THE POPUP MENU USED FOR FAST-WP

An electronic toolkit, known as FAST-WP, was developed at the University of Florida to make it easy for authors and word processors to add special codes to WordPerfect 5.1 (WordPerfect Corporation, 1993a) files running under DOS (Cilley and Watson, 1992a and 1992b). The special codes, WordPerfect styles with generic stylenames, were used to define structural areas within FCES publications. The styles were placed in FCES publications via a pop-up menu. This appendix provides a graphic of the contents of the popup menu.

File	Headings	Mark-up	Notes	Graphics	Symbols
Initialize	Title	Abstract	Edit Footnote 1 ->	Table ->	Bullet ->
View document	Author	Bibliography (entire)	Edit Footnote 2	Figure ->	Cents
Print document	Heading level 1	Bibliographic entry	Add disclaimer ->	Text box ->	Checkmark
Update styles	Heading level 2	Caption	Add caution note ->	Equation ->	Copyright
About FAST-WP	Heading level 3	Emphasis ->	Color plates note		Dagger
	Heading level 4	Footnote in table	Endnote, create		Degree
	Paragraph head	Hyperlink, create ->	Edit endnote		Double dagger
	Fixed case	Hyperlink, edit			Feet
	Edit header	Item in list			Female
		List type ->			Greek letters ->
		Reference title			Inches
		Scientific name			Male
		Subscript			Math menu ->
		Superscript			Punctuation ->
					Registered (R)
					Spanish letters ->

Tools	Setup
Columns on	Install printer
Columns off	Edit printer
Hyphenation ->	Edit WP setup
Page orientation ->	
Reset screen	
Start new column	
Start new page	

APPENDIX G
RELATIONSHIP BETWEEN STYLES AND ELEMENTS

This appendix provides a table describing the relationships between the model elements and their generic styles that were placed in FCES publications using FAST-WP authoring tools.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: (1)	Notes
article	Start of document/end of document	All SGML documents are considered articles. The <article> start tag is the first thing written to the output file, right after it is opened. The </article> end tag is the last thing written to the output file, right before it is closed. Thus, the article tags enclose everything in the SGML file regardless of the content of the WordPerfect file.
fm	Start of document/first text not specifically part of front matter (may include Paired [Style On:Head 1])	The fm element contains the "front matter" of the article, including the title, author, and publication information. The <fm> start tag is written to the output file right after the <article> start tag is written. Thus, the <fm> start tag, like the article tags, is written regardless of the content of the WordPerfect file. The </fm> end tag is written just before the first section of text is opened up. This happens when any text not part of the title, author, or other front matter component is encountered.
bdy	First text not part of front matter (may include Paired [Style On:Head 1])	The <bdy> start tag immediately follows the </fm> end tag, and immediately precedes the opening of the first section of text. The </bdy> end tag is written to the output file right before the </article> end tag, and thus is written regardless of the content of the WordPerfect file.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: **(1)**	Notes
fig	[Fig Box:1;;[Open Style:CaptionO] ...]	Figures are basically ignored by WP2SGML, but the goal is to use the presence of a WordPerfect figure, as above indicated to the left, to place the tags, and to extract information from the figure box to fill information between the tags. Presently, <fig> is empty in the DTD, but we want to add things like caption and filename.
fn	[Footnote:1; ...] or [Endnote:1; ...]	An endnote is created in WordPerfect documents because of the two column format. Our SGML documents contain only footnotes, however, so both footnotes and endnotes are converted using the footnote tags. The <fn> tag is placed in the text corresponding to where the [Footnote] is found in the WordPerfect document. Then the text of the footnote is extracted and placed right after the <fn> start tag. Finally, the text is followed by the </fn> end tag.
hyp	Paired [Style On:Hyperlink]	The <hyp> start tag and </hyp> end tag are placed in the SGML file at the same location in the text where the Hyperlink open and close styles are found in the WordPerfect file. Other components of the hyperword are in a comment which is part of the WordPerfect style. These components are placed where appropriate, inside the correct tags, in the SGML file. Alternatively, the <hyp> and </hyp> styles can be placed in with the help of a specially-formatted Cross-reference file, which WP2SGML uses to place hyperwords not place in the WordPerfect by the author. WP2SGML looks for specified text and places the hyp tags around it.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: **(1)**	Notes
tig	Paired [Style On:Title]	This tag is in our DTD for AAP compatibility. The article title tag, atl, must fall inside the title group, tig, tags. The <tig> start tag replaces the Title style on token in WordPerfect. The </tig> tag replaces the Title style off.
au	Paired [Style On:Author]	Surname is used as tag because it is a required AAP element. At present, the entire author's name is placed in the surname. Later software can be added to parse the names, and tags for first and middle names be added to the DTD.
pubfm	?	This element is written to the output file just before the end of the front matter (</fm>). Thus, the placing of both <pubfm> and </pubfm> depend on the placing of the first section of text, which may or may not be designated by a header style, as explained above. The contents of the style are not taken from text at the location of the start of the section, but are extracted from earlier in the document, or from a cross-reference or XRF file. In the document, contents of the style PubNo and PubDate are saved for placement within the pubfm tags.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: **(1)**	Notes
abs	Paired [Style On:Abstract]	If the very first heading in the document contains the text "Abstract", then the text of that section is placed in the front matter, as the last element of the front matter. The <abs> start tag is placed before this text, and the </abs> end tag is placed after this text. The "Abstract" style should also designate an abstract, but that is yet to be implemented. If there is an abstract, the pubfm (publisher's front matter) comes afterwards.
atl	Paired [Style On:Title]	The same routine that places the tig tags, mentioned above, also places the atl tags. The <atl> and </atl> tags surround the article title, and the tig tags surround the atl tags. A bit redundant for our DTD, but a small price to pay for AAP compatibility.
it	Paired [ITALC] Paired [italc]	Text surrounded by [ITALIC] and [italic] in the WordPerfect file ends up surrounded by <it> and </it> in the SGML file.
b	Paired [BOLD] Paired [bold]	Text surrounded by [BOLD] and [bold] in the WordPerfect file ends up surrounded by and in the SGML file.
el	Paired [Style On:Emphasis]	The <el> start tag is placed in the SGML file where the "Emphasis" style on token appears in the WordPerfect file. At present, FAST-WP does not use "Emphasis", but, if use ever resumes, WP2SGML is ready.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: (1)	Notes
e2	Paired [Style On:Scien Name] or Paired [Style On: ScienName]	Scientific names are enclosed within the <e2> and </e2> tags. Different versions of FAST-WP have different versions of the Scientific Name style name. One is "Scien Name" and one is "ScienName". Both work with WP2SGML.
e3	Paired [Style On:RefTitle]	Reference titles are enclosed within the <e3> and </e3> tags. Different versions of FAST-WP have different versions of the Reference Title style name. One is "Ref. Title" and one is "Reference". Both work with WP2SGML.
f	Paired [SUPRSCTP]/[SUB SCTPT]	For compatibility with the AAP math DTD, superscript (sup) and subscript (inf) tags are enclosed in the formula (f) tag. The formula tag surrounds the superscript and subscript tags within text.
fgr	?	Not used.
sup	Paired [SUPRSCPT][supr scpt]	Superscript tags surround the text within the superscript attribute token. The sup tags are surrounded by f tags.
inf	Paired [SUBSCPT][subsc pt]	Subscript tags surround the text within the subscript attribute token. The sub tags are surrounded by f tags.
snm	Paired [Style On:Author]	Surname is the only AAP author element that is required.
onm	?	Not currently being tagged.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: **(1)**	Notes
crt	?	The Copyright Notice element (crt) is part of the AAP Publisher's Front Matter (pubfm) element. The crt tags surround crd (Copyright Date) tags, which, in turn, surround the Year (yr) tags. Again, this multiple level of tagging is for AAP compatibility. The publication date is placed in the middle of all these tags. This publication date information is extracted from earlier in the WordPerfect document, inside the PubDate style, or is taken from a cross-reference (XRF) file. In either case, it is held until the pubfm element is written to the SGML file.
avl	?	Not currently being tagged.
aid	?	The Article Identification (aid) is part of the AAP Publisher's Front Matter (pubfm) element. In our SGML documents, the aid tags surround the publication number of the document. The publication number is extracted from earlier in the WordPerfect document, inside the PubNo stile, or is taken from a cross-reference (XRF) file. It is held until the pubfm element is written to the SGML file.
issn	?	Not currently being tagged.
crd	?	The Copyright Date (crd) element is part of the crt element, and is what we use to contain publication date of the article. The extracted date is placed inside the crd tags, and also within the yr tags. The crt element, in which the crd element is contained, is part of the pubfm element.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: **(1)**	Notes
mo	?	Not currently being tagged.
day	?	Not currently being tagged.
yr	?	Just as the entire author's name is written between the crt styles, the entire publication date is written between the yr styles. We may try, someday, to parse the date and use the mo and day styles, as well, but, for now, the publication date is placed inside the yr element. The yr element is part of the crd element, which is part of the crt element, which is part of the pubfm element.
h	Paired [Style On:Head, para.] / [Style On:Abstract] / [Style On:Head 5]	This element is used in two different places: in the paragraph head (top1 in SGML, "Head, Para." or "Head 5" in WordPerfect), and in the abstract (abs). In the case of the paragraph head, the text within the style is placed within the h tags, and the entire header, including h tags and the text until another header appears, is placed within the top1 tags. In the case of the abstract, the determined contents of the abstract are placed within the abs tags, and the initial header in the abstract, regardless of its level, is placed within the h tags.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: **(1)**	Notes
p	?	In WP2SGML, flags are set to keep track of when a paragraph is pending, and when paragraphs are allowed. If a paragraph is allowed and not pending, the paragraph is started. Any sentence "terminal" character ('.', '?', or '!'), combined with a hard return or hard page break, ends a paragraph. In some elements, such as list item (li) or table cells (c or cte), paragraphs are required. In these cases, a paragraph is automatically started at the beginning of the element, and automatically ended (if pending) at the end.
tbl	[Tbl Def:...] to [Tbl Off]	A simple table is a table with straightforward structure -- one cell for each row and column -- no more and no less. WP2SGML scans through each table, before scanning, to find out if it is simple or complex. If there are no joined cells, and there are no header lines, the table is a simple table, and is tagged like one. Otherwise, the table is a complex table. A simple table is different from a complex table not only in how it is tagged outside, but in how it its rows and columns are tagged.
ctbl	[Tbl Def:...] to [Tbl Off]	Any table that is not a simple table is a complex table. In WordPerfect, simple and complex tables are marked the same -- it is purely structure that determines the difference between simple and complex tables. In the SGML document, however, the complex table has an entirely different set of tags from the simple table.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: (1)	Notes
l1	Paired [Style On:List I]	Bulleted lists are designated in WordPerfect by the paired "List I" style. The contents of the style are surrounded by l1 tags. In addition, the list header and items are marked appropriately within the list.
l2	Paired [Style On:List U]	Unmarked lists are designated in WordPerfect by the paired "List U" style. The contents of the style are surrounded by l2 tags. In addition, the list header and items are marked appropriately within the list.
l3	Paired [Style On:List E]	Enumerated lists are designated in WordPerfect by the paired "List E" style. The contents of the style are surrounded by l3 tags. In addition, the list header and items are marked appropriately within the list.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: (1)	Notes
sec	Paired [Style On:Head 1] or [Head 2, 3, or 4] or first text in document	The sec tags enclose the highest level sections. Most commonly, the "Head 1" style designates the start of a section. The "Head 1" style is paired, but it contains only the section header, and thus does not serve to place the end tag of the section it is starting. It does, however, serve to place the end tag of any section already pending. The end of the document also tells us where to close the last section. In the ideal document, this would be the end of the story, but some documents have text before the first section header. In this case, a section, with a blank section header, is opened right at the start of the document. It is also possible for the first header, right at the beginning of the document, to be something other than "Head 1". If this header is "Head 2", "Head 3", or "Head 4", that header level will be treated as if it were "Head 1", and "sec" will enclose the sections.
st	Paired [Style On:Head 1, 2, 3 or 4]	All sections and subsections (sec, ss1, ss2, and ss3) contain a section title (st). The contents of the st are whatever the heading style contains. All of the heading levels produce the same st tags. The information about heading level is transferred to section level.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: **(1)**	Notes
ss1	Paired [Style On:Head 2] or [Head 3 or 4]	In the most common case, the "Head 2" style designates the beginning of subsection 1 (ss1). The end of a subsection ss1 is located before the beginning of another ss1, or before the end of a section (sec), whichever comes first. If a "Head 3" or "Head 4" is encountered in the WordPerfect file when a subsection 1 is not pending, then the "Head 3" or "Head 4" is treated as if it were "Head 2", and a new subsection 1 is created.
ss2	Paired [Style On:Head 3] or [Head 4]	In the most common case, the "Head 3" style designates the beginning of subsection 2 (ss2). The end of a subsection ss2 is located before the beginning of another ss2, before the end of a subsection 1 (ss1), or before the end of a section (sec), whichever comes first. If a "Head 4" is in the WordPerfect file when a subsection 2 is not pending (but a subsection 1 is), then the "Head 4" is treated as if it were "Head 3", which creates a new subsection 2.
ss3	Paired [Style On:Head 4]	The "Head 4" style designates the beginning of subsection 3 (ss3). The end of a subsection ss3 is located before the beginning of another ss3, before the end of a subsection 2 (ss2), before the end of a subsection 1 (ss1), or before the end of a section (sec), whichever comes first.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: **(1)**	Notes
top1	Paired [Style On:Head, para.] / [Style On:Head 5]	The "topical paragraph" is treated in much the same way as the sections and subsections. Again, just the header is marked by the style ("Head, para." or "Head 5"). The contents of the topical paragraph begin at the header, and continue until another topical paragraph starts, or until any section level ends.
lh	Paired [Style On:List I] or [Style On:List U] or [Style On:List E]	The list header is everything within a list, between the beginning of the list and the first list item. In the SGML file, the list tag (l1, l2, l3) is placed, followed by the lh tag, followed by any text that follows before the first "List item" style. If there is no list item, then everything within the list is the list header, and ends up between the lh tags.
li	Open [Style On:List item]	The list item style is an open style. This means that there is a style marking its beginning, but nothing to mark its end. The li tag, however, is paired. The tag is placed where the "List item" style is found. The end tag is placed before the next tag, or at the end of the list, whichever comes first.
lword	Paired [Style On:Hyperlink]	The link word (lword) is the text contained in the "Hyperlink" style. The text in that style is placed inside the lword tags, which, in turn, are contained in the hyp (hyperword) tags.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: (1)	Notes
act	Paired [Style On:Hyperlink]	The action for a hyperword is generally a filename. Whatever it is, it indicates what is to be done when the hyperword is chosen. In the WordPerfect document, it is found inside the Hyperlink style, as part of a comment inside the text. The action is found after the words, "Link filename:" within the comment. It is extracted until a hard return is encountered, and placed within the act tags, which are inside the hyp tags. The <act> start tag also has some attributes. The values for these attributes are found within the same WordPerfect comment that the action is found in. The value for the "type" attribute is found after the words "Link type:" and before the first hard return. The "type" attribute gives information on what is to be done with the "action". Finally, there is a description of the hyperlink, to make it clear to human readers what the hyperlink is supposed to do. This description is found in the SGML file as the value of the "descr" attribute. The text for this value is found in the comment between the word "Description:" and the first hard return.
no	?	Table numbers apply to both simple and complex tables. We are not using table numbers at present.
tt	Paired [CaptionO] or first table row	The captions of a table box, the first header row of a table if joined into one cell, and the marked title of a table outside the table itself are all proposed methods of marking a table title. Currently, WP2SGML doesn't process them.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: **(1)**	Notes
tby	?	This is the simple table body. It follows any header rows (A simple table can have two), and contains all of the normal rows in a table. The start tag <tby> is placed right after the first header row, and the end tag </tby> is placed right before the end of the table (</tbl>)
row	Open [Row]	This marks the row in a simple table. A start tag for a row (<row>) is placed wherever a "Row" marker appears in the WordPerfect text. The end tag </row> is placed before the beginning of any new row (other than the first), and also just before the end of the table body </tby>.
th	Open [Cell]	The Table Column Header style (th) can be used for the cells in the first header row in a simple table. There is no header row element, but, for the th element to be meaningful, there must be one set of th tags for each column in the table. Table Column Header start tags (<th>) are placed in the text where the "Cell" markers are in the first header row. The end tags (</th>) are placed before subsequent start tags, and at the end of the first header row. At present, all tables with header rows are deemed complex tables, so this element is not tagged.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: (1)	Notes
tsh	Open [Cell]	The Table Column Subordinate Header style (tsh) can be used for the cells in the second header row in a simple table. There is no header row element, but, for the tsh element to be meaningful, there must be one set of tsh tags for each column in the table. Table Subordinate Column Header start tags (<tsh>) are placed in the text where the "Cell" markers are in the first second header row. The end tags (</tsh>) are placed before subsequent start tags, and at the end of the second header row. At present, all tables with header rows are deemed complex tables, so this element is not tagged.
tsb	Open [Row]	The table stub line (tsb) can be used as the first cell of a row in simple tables. If this option is used, then the first cell marker is where the start tag <tsb> is placed, and the next "Cell" marker is where the end tag </tsb> and the first open cell start tag <c> is placed. Currently, cells are marked as cells and there are no table stub lines.
c	Open [Cell]	Most "Cell" markers in a simple table will be marked with the c tags. The only exceptions are header cells (th and tsh), and table stub lines (tsb). Since neither of these is presently implemented, all cells are presently marked with c tags. The <c> start tag is place before every "Cell" marker in a row. The end tag </c> is placed before every <c> tag except the first, and also before the </row> end tag.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: **(1)**	Notes
cthd	[Tbl Def:...]	The Complex Table Head is placed at the beginning of the any complex table. Optionally, it contains the table number (not inserted by WP2SGML), the complex table title (ctt -- not extracted by WP2SGML), the table dimensions (tdim), and the Complex Table Header Rows (cthr). The <cthd> start tag is placed at the beginning of the table, right after the <ctbl> start tag. The </cthd> end tag is placed right after the last </cthr> end tag.
ctby	?	The Complex Table Body (ctby) contains all of the normal rows in a complex table -- that is, all of the rows that are not header rows. A WordPerfect table has, as one of its defining characteristics, the number of lines there are in the header. After the last line of header row (cthr), the </cthd> end tag is placed, as stated above. Right after that, the <ctby> start tag is placed. The </ctby> end tag is placed at the end of the table, just before the Complex Table Foot start tag (<ctbf>).
ctbf	[Tbl Off]	The Complex Table Foot is found at the end of a table. Both the start tag <ctbf> and the end tag </ctbf> are placed where the "Table Off" marker is found in the WordPerfect. Right now, the ctbf is not used for anything. It is there for AAP compatibility. Inside the ctbf tags, is placed a Complex Table Caption (ctc) which, in turn, contains an empty paragraph (p).

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: **(1)**	Notes
ctt	Paired [CaptionO] or first table row	The captions of a table box, the first header row of a table if it is joined into one cell, and the marked title of a table outside the table itself are all proposed methods of marking a table title. Presently, none of them is performed by WP2SGML
tdim	?	This is a non-AAP addition to our DTD which gives rectangular dimensions to every cell in a table. Its main purpose is to give an idea of proportionality, so the retrieval software can represent the table in roughly the same way as it originally appeared. It is placed before the table body (tby) in simple tables, and in the complex table header (cthd), just before the complex table header rows (cthr), in complex tables. The initial scan of tables, to find out if they are simple or complex, also serves to gather these dimensions, which are then placed within the colwid and rowhgt tags.
cthr	Open [Row]	WordPerfect tables have indicators for header rows. Other than this indicator, there is nothing to say which rows are the header rows. Each row lucky enough to be a header row is placed inside the complex table header (cthd).

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: (1)	Notes
colwid	?	Our DTD says this non-AAP element contains #PCDATA, so it could be about anything. However, right now, it is a comma-delimited set of column widths in 1200^{ths} of an inch. That is, divide the number by 1200, and you have inches. Every column in the table is represented, even if it is just a split cell in one row. The widths are listed from left to right. The colwid element is part of the tdim element.
rowhgt	?	Our DTD says this non-AAP element contains #PCDATA, so it could be about anything. However, right now, it is a comma-delimited set of row heights in 1200^{ths} of an inch. That is, divide the number by 1200, and you have inches. Every row in the table is represented, even if it is just a split cell in one column. The heights are listed from top to bottom. The rowhgt element is part of the tdim element.
cth	Open [Cell]	The complex table header is a cell in a complex table header row. As with other cells, both in simple tables and in complex tables, the start tag <cth> is placed in the text where the "Cell" marker appears in the WordPerfect document, and the end tag </cth> is placed before all <cth> start tags except the first, and also before the end of the row.

The Relationship between DTD Elements and WordPerfect Styles for FCES Documents.

DTD Element	Identified in WP by: (1)	Notes
ctsbl	Open [Cell]	The complex table stub line (ctsbl) can be used as the first cell of a row in complex tables. If this option is used, then the first cell marker is where the start tag <ctsbl> is placed, and the next "Cell" marker is where the end tag </ctsbl> and the first open cell start tag <cte> is placed. Right now, all cells are marked as cells, and there are no complex table stub lines.
cte	Open [Cell]	Most "Cell" markers in a complex table will be marked with the cte tags. The only exceptions are header cells (cth), and table stub lines (ctsbl). The stub lines are not implemented, so every cell outside the header is marked with cte tags. The <cte> start tag is place before every "Cell" marker in a row. The end tag </cte> precedes each <cte> tag except the first, and also before the </ctr> end tag.
ctc	?	The complex table caption will likely never represent anything, since, if there is a caption to a table box, it will probably be placed as the ctt, the complex table title. Perhaps, if there is a title in addition to the caption, the caption will be placed in ctc.
ctr	Open [Row]	This marks the row in a complex table. A start tag for a row (<ctr>) is placed wherever a "Row" marker appears in the WordPerfect text. The end tag </ctr> is placed before the beginning of any new row (other than the first), and also just before the end of the table body </tby>.

1) ? refers to element not identified in WordPerfect.

APPENDIX H
LITERATURE REVIEWED BUT NOT INCLUDED IN THESIS

Barnard, D.T., Fraser, C.A., and Logan, G.M., Generalized Markup for Literary Texts. Literary and Linguistic Computing, (3)1:26-31, 1988. Oxford University Press.

Franklin, C., A Bibliography on Hypertext and Hypermedia with Selected Annotations. Database 13(1):24-32, February, 1990.

Gangemi, J.V., The AAP Electronic Manuscript Project - The Typesetter's Perspective. Electron. Publ. Bus. 4(8):19, September, 1986. CODEN: EPBUEG, ISSN: 0888-0948.

Goldstein, C.M., Full Text Retrieval from Structured Text. Bulletin of the American Society for Information Science page 11, August/September, 1989.

Guting, R.H., Zicari, R., and Choy, D.M., An Algebra for Structured Office Documents. ACM Transactions on Information Systems 7(2):123-157, April, 1989.

Habermann, A.N., and Notkin, D., Gandalf: Software Development Environments. IEEE Transactions on Software Engineering 12(12):1117-1127, December, 1986.

Hunter, K., AAP/SGML - Implications for Journal Publishers. Electronic Publishing Business, 4(8):16-17, September 1986. Elsevier Science Publishers, New York, NY.

Kircz, J.G., and Bleeker, J., The Use of Relational Databases for Electronic and Conventional Scientific Publishing. Journal of Information Science 13:75-89, 1987. CODEN: JISCDI, ISSN:0165-5515.

Klensin, J.C., INFOODS Food Composition Data Interchange Handbook. Food and Nutrition Bulletin, Supplement. The United Nations University, Tokyo, 1990. 145 pages.

Lancashire, I., and McCarty, W., Text Encoding and Enrichment. The Humanities Computing Yearbook 1988. Pages 343-345 (section 25.10). Clarendon Press, Oxford, 1988.

Martin, J.S., Electronic Document Interchange and The AAP Electronic Manuscript Project. Library Hi Tech 4(3):31-42, Fall 1986. The Information Society, Inc..

Meyrowitz, N., and van Dam, A., Interactive Editing Systems. ACM Computing Surveys 14(3):321-415, 1982.

News and Notes. Mellon Foundation Supports International Text Encoding Project with $100,000 Grant. Literary and Linguistic Computing 5(1):108-109, 1990.

Ossher, H., A case study in structure specification: A grid description of Scribe. IEEE Transactions on Software Engineering 15(11):1397-1416, November, 1989. ISSN: 0098-5589.

Painter, J.D., Marking up the Dictionary (The Oxford Dictionary). Information Media & Technology 21(2):72-74, March, 1988. CODEN: IMTEED, ISSN: 0306-2880.

Morris, J.H., Satyanarayanan, M., Conner, M.H., Howard, J.H., Rosenthal, D.S.H., and Smith, F.D., Andrew: A Distributed Personal Computing Environment. Communications of The ACM 29(3):184-201, 1986.

Morris, R., Rendering Digital Type: A Historical and Economic View of Technology. The Computer Journal 32(6):524-532, 1989.

Smith, J.M., International Standards for the Interchange of Text. Oxford Surveys in Information Technology 2:165-194, 1985.

Smith, J.M., The Implications of SGML for the Preparation of Scientific Publications. Computer Journal 29(3):193-200, June, 1986. CODEN: CMPJA6, ISSN: 0010-4620.

Smith, J.M., The Standard Generalized Markup Language (SGML) for Humanities Publishing. Literary and Linguistic Computing 2(3):171-175, 1987. CODEN: LLCOEI, ISSN: 0268-1145.

Stutely, R., HMSO and Generic Coding - Past, Present, Future. Information Media & Technology 20(1):18-20, January, 1987. CODEN: IMTEED, ISSN: 0306-2880.

Tice, G., Document Standard Focuses on the Larger Picture. IEEE Software 6(5):80, 82, September, 1989. ISSN: 0740-7459.

van Herwijnen, E., The Use of Text Interchange Standards for Submitting Physics Articles to Journals. Computer Physics Communications 57:244-250, 1989.

REFERENCE LIST

Achugbue, J.O., On the Line Breaking Problem in Text Formatting. ACM SIGPLAN Notices 16(6):117-122, 1981.

ACM Press Database and Electronic Products - New Services for the Information Age. Communications of the Association for Computing Machinery 31(8):948-951, 1988.

Agfa, CAPS 6.0 Software. Wilmington, MA, 1991.

Akscyn, R., McCracken, D.L., and Yoder, E., KMS: A Distributed Hypertext for Sharing Knowledge in Organizations. Communications of the ACM 31(7):820-835, 1988.

Allen, T., Nix, R., and Perlis, A., PEN: A Hierarchical Document Editor. ACM SIGPLAN Notices 16(6):74-81, 1981.

American National Standards Institute, Standard for electronic manuscript preparation and markup. ANSI/NISO Z39.59, Washington, DC, 1988.

Ansen, D., Document Architecture Standards Evolution: Office Document Architecture for Structuring and Encoding Documents. AT & T Technical Journal 68(4):33-55, 1989.

Apple Computer, Inc., MacWrite. Cupertino, CA, 1984.

ArborText, Inc., Adept Publisher, Document Architect, Adept Editor, and Adept Power Paste. Ann Arbor, MI, 1994.

Association of American Publishers, The AAP Article DTD. The Electronic Publishing Special Interest Group (EPSIG), Dublin, OH, 1987.

auto-graphics, inc., SGML Smart Editor. Pomona, CA 91768. 1994.

Avalanche Development Company, FastTag, SGML Hammer, Interleaf Developer's Kit, SGML Developer's Kit, and SureSTYLE. Boulder, CO, 1994.

Barnes, J.A., Analysis of Document Coding Schemes: A General Model and Retagging Toolset. Technical Report OSU-CISRC-7/90-TR19, The Ohio State University, July, 1990.

Beck, H.W., Gala, S., and Navathe, S., Classification as a Query Processing Technique in the Candide Semantic Data Model. Proceedings, IEEE Fifth International Conference on Data Engineering, Los Angeles, CA. 1989a.

Beck, H.W., Jones, P., Watson, D.G., and Zazueta, F., An Expert Database System for Ornamental Plants. Agricultural Systems, Elsevier Science Publishers Ltd, England, 31:111-126, 1989b.

Beck, H.W., and Watson, D.G., Analysis of Agricultural Extension Documents. AI Applications, 6(3):17-28, 1992.

Beck, H.W., Jones, P., and Watson, D.G., A CD-ROM Based Agricultural Information Retrieval System. Applied Engineering in Agriculture, 10(1):127-132, 1994.

Brown, P.J., Turning ideas into products: The Guide System. Proceedings of Hypertext Workshop, University of North Carolina, Chapel Hill, NC, 1987.

Bryan, M., SGML: An Author's Guide to the Standard Generalized Markup Language. Addison-Wesley Publishing Company, Reading, MA, 1988.

Bush, V., As we may think. The Atlantic Monthly 176(1):101-108, 1945.

Carmody, S., Gross, W., Nelson, T., Rice, D., and van Dam, A. A hypertext editing system for the /360. Pertinent Concepts in Computer Graphics, University of Illinois Press, Urbana, IL, pp. 291-330, 1969.

CD-ROM Implementation Group, Authoring CD-ROM Systems for Extension Information Delivery. The University of Florida, Lake Buena Vista, FL, 1990.

Chamberlin, D.C., King, J.C., Slutz, D.R., Todd, S.J.P., and Wade, B.W., JANUS: An interactive system for document composition. Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation, SIGPLAN Notices 16(6):82-91, 1981.

Chamberlin, D.C., King, J.C., Slutz, D.R., Todd, S.J.P., and Wade, B.W., JANUS: An Interactive Document Formatter Based on Declarative Tags. IBM Computer Science Research Report RJ3366 (40402), 1982.

Cilley, M.L., Jones, P.H., Watson, D.G., Zazueta, F.S., and Beck, H.W., Using WordPerfect to Implement SGML. DISC Magazine. 1(1):48-51, 1990.

Cilley, M.L., and Watson, D.G., FAST-WP: Florida's Authoring System Tools for WordPerfect Version 3.1 User's Guide. Florida Cooperative Extension Service Circular 1069, University of Florida, Gainesville, Fl, 1992a.

Cilley, M.L., and Watson, D.G., FAST-WP Developers' Guide. Florida Cooperative Extension Service Circular 1083, University of Florida, Gainesville, Fl, 1992b.

Conklin, J., Hypertext: A survey and introduction. IEEE Computer 20(9):17-41, 1987.

Coombs, J.H., Renear, A.H., and DeRose, S.J., Markup systems and the Future of Scholarly Text Processing. Communications of the ACM 30(11):933(15), 1987.

Cover, R. 1992. Standard Generalized Markup Language (ISO 8879:1986 SGML:Annotated Bibliography and List of Resources. University of Exeter, United Kingdom.

Cover, R., Duncan, N., and Barnard, D.T., February 1991. Bibliography on SGML (Standard Generalized Markup Language) and Related Issues. Technical Report 91-299. Department of Computer and Information Science, Kingston, Ontario, Canada.

Data Conversion Laboratory, MindReader and SGMLword. Fresh Meadows, NY, 1994.

Datalogics, WriterStation and ParseStation, Chicago, IL, 1991.

Day & Zimmermann, SGML Import Program and Interactive Presentation Manager, Middletown, RI, 1994.

Delisle, N., and Schwartz, M., Neptune: A Hypertext System for CAD Applications. Proceedings of ACM SIGMOD '86, ACM, New York, pp. 132-142, 1986.

Delisle, N., and Schwartz, M., Contexts - a partitioning concept for hypertext. Proceedings of the Conference on Computer-Supported Cooperative Work (Austin, TX, December 3-5), ACM, New York, pp. 147-153, 1986.

Delisle, N., and Schwartz, M., Contexts - a partitioning concept for hypertext. ACM Transactions on Office Information Systems 5(2):168-186, 1987.

Electronic Book Technologies, Inc., DynaText and DynaTag, Providence, RI, 1994.

Englebart, D.C., and English, W., A research center for augmenting human intellect. Proceedings of 1968 FJCC (San Francisco, CA December 9-11) AFIPS Press, Montvale, NJ, pp. 395-410, 1968.

Englebart, D.C., Watson, R.W., and Norton, C., The augmented knowledge workshop. ARC Journal Accession Number 14724, Danford Research Center, Menlo Park, CA, 1973.

Englebart, D.C., Authorship provisions in Augment. Proceedings of the IEEE COMPCON (San Francisco, CA, Spring), IEEE, New York, pp. 465-472, 1984.

Exoterica Corporation, XGML Application Developer's Reference Manual. Ottawa, Ontario Canada, 1987.

Exoterica Corporation, OmniMark. Ottawa, Ontario Canada, 1994.

Feiner, S., Nagy, S., and van Dam, A., An Integrated System for Creating and Presenting Complex Computer-Based Documents. Computer Graphics 15(3):181-189, 1981.

Feiner, S., Nagy, S., and van Dam, A., An Experimental System for Creating and Presenting Interactive Graphical Documents. ACM Transactions on Graphics 1(1):59-77, 1982.

Folio Corporation, Folio VIEWS SGML Toolkit. Provo, UT, 1994.

Francis, B., A Corporate Vision for Publishing. Datamation 36(2):65-67, 1990.

Frisse, M.E., From Text to Hypertext. Byte 13(10):247-253, 1988.

Frisse, M.E., Searching for Information in a Hypertext Medical Handbook. Communications of the ACM 31(7):880-886, 1988.

Furuta, R., An Object-based Taxonomy for Abstract Structures in Document Models. Computer Journal 32(6);494-504, 1989.

Furuta, R., Scofield, J., and Shaw, A., Document Formatting Systems: Survey, Concepts, and Issues. ACM Computing Surveys 14(3):417-472, 1982.

Ganzinger, H., and Wilmertinger, W., FOAM: A Two-Level Approach to Text Formatting on a Microcomputer System. Software - Practice and Experience 15(4):327-342, 1985.

Garg, P.K., and Scacchi, W., A Hypertext System to Manage Software Life-cycle Documents. IEEE Software 7(3):90-98, 1990.

Garrett, L.N., Smith, K.E., and Myrowitz, N., Intermedia: Issues, Strategies, and Tactics in the design of a hypermedia document system. Proceedings of the Conference on Computer-Supported Cooperative Work (Austin, TX, December 3-5), ACM, New York, pp. 163-174, 1986.

Goldfarb, C.F., Document Composition Facility Generalized Markup Language: Concepts and Design Guide, Form Number SH20-9188-0, IBM Corporation, White Plains, NY, 1980.

Goldfarb, C.F., The SGML Handbook. Oxford University Press, New York, 1990.

Goldfarb, C.F., Mosher, E.J., and Peterson, T.I., An Online System for Integrated Text Processing. Proceedings of the American Society for Information Science 7:147-150, 1970.

Goodstein, D.H., and Cameron, T.L., Space-age Electronic Publishing. Computer Graphics World 11:41-48, 1988.

Graphic Communications Association, Basic SGML Tutorial. Dallas, TX, April 22-25, 1991.

Grif S.A., Grif SGML Editor, Quentin en Yvelines, France, 1994.

Gutknecht, J., Concepts of the Text Editor Lara. Communications of the ACM 28(9):942-960, 1985.

Gutknecht, J., and Winiger, W., Andra: The Document Preparation System of the Personal Workstation Lilith. Software - Practice and Experience 14:73-100, 1984.

Haggerty, M., NCGA and 'Real Solutions' in California. [National Computer Graphics Association's 11th Annual Conference in Anaheim, CA, IEEE Computer Graphics and Applications 10(2):8-11, 1990.

Halasz, F.G., Moran, T.P., and Trigg, R.H., Notecards in a Nutshell. Proceedings of the 1987 ACM Conference of Human Factors in Computer Systems (CHI+GI 1987), Toronto, Ontario, Canada, pp. 45-52, April 5-9, 1987.

Halasz, F.G., Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems. Communications of

the Association for Computing Machinery 31(7):836-852, 1988.

Hammer, M., Ilson, R., Anderson, T., Gilbert, E.J., Good, M., Niamir, B., Rosenstein, L., and Schoichet, S., Etude: An integrated document processing system. Office Automation Group Memo OAM-028, M.I.T. Laboratory for Computer Science, Cambridge, MA, February 1981.

Harrison, T.V., Watson, D.G., Beck, H.W., Cilley, M.L., and Eissinger, S.T., Modelling Documents For Automated Knowledge Base Entry, Winter Meeting in Nashville, TN, Paper Number 923503, American Society of Agricultural Engineers, St. Joseph, MI, December 15-18, 1992.

Horak, W., Concepts of the Document Interchange Protocol for the Telematic Services - CCITT Draft Recommendation S.a. Computer Networks 8:175-185, 1984.

Ilson, R., An integrated approach to formatted document production. Technical Report MIT/LCS/TR-253, M.I.T. Laboratory for Computer Science, Masters Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1980.

Ilson, R., Interactive effectivity control: Design and applications. Proceedings of ACM Conference on Document Processing Systems, Santa Fe, NM, ACM, New York, pages 85-91, December 5-9, 1988.

InContext Systems, InContext 2. Bethesda, MD, 1994.

Information Dimensions, Basis SGMLserver. Dublin, OH, 1994.

InfoAccess, Inc., Guide Professional Publisher. Bellevue, WA, 1994.

Interleaf, Inc., Interleaf5<SGML> and Toolkit. Waltham, MA, 1994.

International Standards Organization, Information Processing-Text and Office Systems-Standard Generalized Markup Language (SGML). ISO 8879-1986, Geneva, Switzerland, 1986.

Janssen, N.J.M., Nawijn, W., Nijland, T.L.B., and Peels, A.J.H.M., Principles of operation of the COBATEF multiprocessor system. Rep. INF85-7, Department of Computer Science, Twente University of Technology, Enschede, The Netherlands, June 1985.

Kaebling, M.J., Braced Languages and a Model of Translation for Context-Free Strings: Theory and Practice. Ph.D. Dissertation, The Ohio State University, 1987.

Kaysen, J., Let's Play Tag: The Promise of SGML. OCLC Micro 5(5):26-28, 1989.

Kernighan, B.W., A typesetter-independent TROFF. Computing Science Technical Report 97, Bell Laboratories, Murray Hill, NJ, 1981.

Kerr, S., Document Interchange Reigns at DOD. Datamation 34(23):81-84, 1988.

Kimura, G.D., A Structure Editor for Abstract Document Objects. Ph.D. dissertation, Department of Computer Science, University of Washington, Seattle, WA, July 1984.

Kimura, G.D., A Structure Editor for Abstract Document Objects. IEEE Transactions on Software Engineering SE-12(3):417-435, 1986.

Kimura, G.D., and Shaw, A.C., The Structure of Abstract Document Objects. Proceedings of the Second ACM-SIGOA Conference on Office Information Systems, SIGOA Newsletter 5(1-2):161-169, 1984.

Knuth, D.E., The TEXbook. Addison-Wesley, Reading, MA, 1984.

Lamport, L., LATEX: A Document Preparation system. Addison-Wesley, Reading, MA, 1985.

Lee, J., and Malone, T.W., How Can Groups Communicate When They Use Different Languages? - Translating Between Partially Shared Type Hierarchies. Conference on Office Information Systems (OIS) at Palo Alto, CA, IEEE, pp. 22-29, March 1988.

Lemay, L., Teach Yourself Web Publishing With HTML In A Week. SAMS Publishing, Indianapolis, IN, 1995.

Lexicon Systems, Inc., ESSE. Colorado Springs, CO, 1994.

Linington, P.F., File Transfer Protocols. IEEE Journal on Selected Areas in Communications (J-SAC) 7(7):1052-1059, 1989.

Logan, H.M., Report on a New OED Project: A Study of the History of New Words in the New OED. Computers and The Humanities 23(4-5):385-395, 1989.

Macleod, I.A., Storage and Retrieval of Structured Documents. Information Processing and Management 26(2):197-208, 1990.

Macleod, I.A., and Reuber, A.R., The Array Model: A Conceptual Modeling Approach to Document Retrieval. Journal of the American Society for Information Science 38(3):162-170, 1987.

Malone, T.W., Grant, K.R., Turbak, F.A., Brobst, S.A., and Cohen, M., Intelligent Information-Sharing Systems. Communications of the ACM 30(5):390-402, 1987.

Malone, T.W., Grant, K.R., Lai, K.Y., Rao, R., and Rosenblitt, D., Semi-structured messages are surprisingly useful for computer supported coordination. Transactions on Office Information Systems 5(2):115-131, 1987.

Mamrak, S.A., Kaelbling, M.J., Nicholas, C.K., and Share, M., A Software Architecture for Supporting the Exchange of Electronic Manuscripts. Communications of The ACM 30(5):408-414, 1987.

Mamrak, S.A., Barnes, J., Hong, H., Joseph, C., Kaelbling, M., Nicholas, C., O'Connell, C., and Share, M., Descriptive Markup - The Best Approach? Communications of The ACM 31(7):810-811, 1988a.

Mamrak, S.A., Barnes, J.,Bushek, J., and Nicholas, C.K., Translation between Content-Oriented Text Formatters: Scribe, LaTex and Troff. Ohio State University, Columbus, OH, August 1988b.

Mamrak, S.A., and Joseph, C., Translation for WYSIWYG Word Processors in Chameleon. SIGOIS Bulletin 9(1):14-20, 1988c.

Mamrak, S.A., O'Connell, C.S., Hong, H., and Parent, R., The Automatic Generation of Software for Data Exchange in the Graphics Domain. Technical Report OSU-CISRC-7/88-TR20, The Ohio State University, 1988d.

Mamrak, S.A., Kaelbling, M.J., Nicholas, C.K., and Share, M., Chameleon: A System for Solving the Data-Translation Problem. IEEE Transactions on Software Engineering 15(9):1090-1108, September, 1989.

Mamrak, S.A., O'Connell, C.S., and Parent, R.E., The Automatic Generation of Translation Software for Graphical Objects. IEEE Computer Graphics and Applications 9(6):34-43, 1989.

Mamrak, S.A., O'Connell, C.S., and Barnes, J.A., The Integrated Chameleon Architecture: A Software Toolset to Support Data Translation. Technical Report OSU-CISRC-11/90-TR37, Department of Computer and Information Science, The Ohio State University, 1990.

Marchionini, G., Making the Transition from Print to Electronic Encyclopaedias: Adaptation of Mental Models. International Journal of Man-Machine Studies 30(6):591-618, 1989.

Martin, J., Converting the Printed Word to Machine-Readable Text. PC Week 37(70):70-71, 1990.

McCracken, D.L., and Akscyn, R., Experience with the ZOG human-computer interface system. International Journal of Man-Machine Studies 21(2):293-310, 1984.

McGrath, R., Pretty pages (Software Review: Imsys desktop publishing software). Computer Graphics World 12(8):85-87, 1989.

Meyrowitz, N., Intermedia: The Architecture and Construction of an Object-Oriented Hypertext/Hypermedia System and Applications Framework. Proceedings of the Conference on Object-Oriented Programing Systems, Languages, and Applications (OOPSLA '86, Portland, Oregon), pp. 186-201, September 29-October 2, 1986.

Microsoft Corporation, SGML Author for Word and Multimedia Viewer, Redmond, WA 98052. 1994.

Microstar Software, Ltd., NEAR & FAR. Ontario, Canada, 1994.

MIPS Music Standards Committee. Literary and Linguistic Computing 2(1):59, 1987.

Mylonas, E., and Heath, S., Hypertext from the Data Point of View. In Rizk, A., Streitz, N., and Andre, J., editors, Hypertexts: Concepts, Systems and Applications. Cambridge University Press, Cambridge, NY, 1990.

Nelson, N.M., AAP Sets Things Straight. CD ROM Review 2(2):10-16, 1987.

Nelson, T.H., Managing Immense Storage. Byte 13(1):225-238, 1988.

Newcomb, S.R., Kipp, N.A., and Newcomb, V.T., "HYTIME" Hypermedia/Time-Based Document Structuring Language. Communications of The ACM 34(11):67-83, 1991.

NICE Technologies, SGML TagWizard, Capitola, CA, 1994.

Nicholas, C.K., Assuring Accessibility of Complex Software Systems. Ph.D. Dissertation, The Ohio State University, 1988.

Nicholas, C.K., and Mamrak, S.A., Assuring Accessibility of Complex Software Systems. Technical Report OSU-CISRC-7/88-TR21, Department of Computer and Information Science, The Ohio State University, 1988.

Nielsen, J., The Art of Navigating Through Hypertext. Communications of The ACM 33(3):296-310, 1990.

Nijland, T.L.B., and Peels, A.J.H.M. The COBATEF multiprocessor system -- Hardware description. Rep. INF85-8, Department of Computer Science, Twente University of Technology, Enschede, The Netherlands, June 1985.

O'Connell Jr., C.S., Supporting the development of grammar descriptions for multiple applications. Technical Report OSU-CISRC-7/90-TR20, Department of Computer and Information Science, The Ohio State University, 1990.

Officesmiths, Inc., The Officesmith Markup Language. Ottawa, Ontario, Canada, 1991.

Owl International, Guide Users Manual. Bellevue, WA, 1986.

Painter, J.D., A Layman's View of Document Architecture and Interchange. Information Media and Technology 22(1):18-20, 1989.

Passage Systems, Inc., PassagePRO. Mountain View, CA, 1994.

Peels, A.J.H.M., A design method for microprocessor-based systems. Ph.D. dissertation, Twente University of Technology, Enschede, The Netherlands, 1984.

Peels, A.J.H.M., Janssen, N.J.M., and Nawijn, W., Document Architecture and Text Formatting. ACM Transactions on Office Information Systems 3(4):347-369, 1985.

Pfaffenberger, B., Mosaic User's Guide. MIS Press, New York, NY, 1994.

Price, L.A., Thumb: An Interactive Tool for Accessing and Maintaining Texts. IEEE Transactions on Systems, Man, and Cybernetics, pp. 155-162, March-April, 1982.

Purton, P., Europe's electronic trading bloc. Datamation 35(5):76/13-14, 1989.

Borland International, Inc., Quattro Pro Version 4.0, Scotts Valley, CA, 1992.

Quint, V., and Vatton,I., An Abstract Model for Interactive Pictures. Human-Computer Interaction - INTERACT '87, pp. 643-647, North-Holland, 1987.

Rahtz, S., The Processing of Words. Information Technology in the Humanities: Tools, Techniques and Applications. Ellis Horwood, Chichester, pp. 69-70, 1987.

Raymond, D.R., and Tompa, F.W., Hypertext and the Oxford English Dictionary. Communications of the ACM 31(7):871-879, 1988.

Reid, B.K., Scribe: A Document Specification Language and its Compiler. Technical Report Number CMU-CS-81-100, Carnegie-Mellon University, Pittsburg, PA, 1980.

Rodgers, D.L., CALS calls for uniformity in publishing (Computer-Aided Acquisition and Logistics Support). Government Computer News 8(17):68, 1989.

Robertson, G., McCracken, D.L., and Newell, A., The ZOG Approach to Man-Machine Communication. International Journal of Man-Machine Studies (14):461-488, 1981.

Rubinsky, Y., A Presentation. The Seybold Report on Publishing Systems 18(14):17(9), 1989.

Scheller, A., Document Standards: Availability and Products. Computer Network's and ISDN Systems 16(1-2):138-142, 1988.

Sema Group, MARK-IT AND WRITE-IT. Bel Air, MD, 1991.

SGML Associates, Inc., SGML Bibliography. <TAG> The SGML Newsletter 5(5):33(3), Aurora, CO, 1992.

Shaffstall Corporation, SGML Translator. Indianapolis, IN, 1992.

Share, M., Ambiguous Translation Grammars. Ph.D. Dissertation, The Ohio State University, 1988a.

Share, M., Resolving Ambiguities in the Parsing of Translation Grammars. SIGPLAN Notices 23(8):103-109, 1988b.

Shaw, A.C., A model for document preparation systems. Technical Report 80-04-02, University of Washington, Seattle, WA, April 1980.

Shneiderman, B., User interface design and evaluation for an electronic encyclopedia. Technical Report CS-TR-1819. University of Maryland, College Park, MA, March 1987a.

Shneiderman, B., User interface design for the Hyperties electronic encyclopedia. Proceedings of the Hypertext 1987 Workshop at The University of North Carolina at Chapel Hill, NC, ACM, New York, November 1987b.

Smith, J.M., Language Representation the General Way. Computers in Literary and Linguistic Computing: Proceedings of the Eleventh International Conference, Paris, France, pp. 355-369, 1985.

Smith, J.M., Standards. Literary and Linguistic Computing 1(3):191-192, 1986a.

Smith, J.M., Starting to use the Standard Generalized Markup Language (SGML). Electronic Publishing: Compact Disc & Corporate Publishing. Proceedings of the International Conference, London, UK, pp. 65-74, September 1986b.

Smith, J.M., Standards. Literary and Linguistic Computing 4(1):57-58, 1989a.

Smith, J.M., Standards. Literary and Linguistic Computing 4(4):294-296, 1989b.

SoftQuad Inc., The SGML Primer Second Edition. Toronto, Canada, 1991.

SoftQuad Inc., Author/Editor. Toronto, Canada, 1994.

Stein, M.J., Writing Simultaneously for Hardcopy and Softcopy. Proceedings of the 38th International Technical Communication Conference, The Society for Technical Communication, Arlington, VA, pp. RT37 - RT40, 1991.

Stotts, P.D., and Furuta, R., Adding browsing semantics to the hypertext model. Proceedings of ACM Conference on Document Processing Systems, Santa Fe, NM, pp. 43-50, December 5-9, 1988.

Stotts, P.D., and Furuta, R., Petri net based hypertext: document structure with browsing semantics. ACM Transactions on Information Systems 7(1):3-29, 1989.

Strandberg, J., Chomsky, C., Scholes, R., and van Dam, A., An Experiment in Computer-Based Education Using Hypertext. Division of Applied Mathematics and Department of English, Brown University, Providence, RI, June, 1976.

SGML Associates, Inc., <TAG> The SGML Newsletter. Aurora, CO, May 1992.

The Office of Technology Assessment, Informing the Nation: The Future of Federal Electronic Printing, Publishing, and Dissemination. Congress of the U.S., Washington, DC, 1988.

Teitelman, W., A tour through Cedar. IEEE Software 1(2):44-73, 1984.

Teitelman, W., A tour through Cedar. IEEE Transactions on Software Engineering SE-11(3):285-302, 1985.

Texcel Software GmbH, Information Manager. Mannheim, Germany, 1994.

Tompa, F.W., A Data Model for Flexible Hypertext Database Systems. ACM Transactions on Information Systems 7(1):85-100, 1989.

Trigg, R.H., Guided Tours and Tabletops: Tools for Communicating in a Hypertext Environment. ACM Transactions on Office Information Systems 6(4):398-414, 1988.

Trigg, R.H., and Weiser, M., TEXTNET: A Network Based Approach to Text Handling. ACM Transactions on Office Information Systems 4(1):1-23, 1986.

Unilogic, Ltd, Scribe Introductory User's Manual Fourth Edition. Pittsburgh, PA, April 1984.

US Department of Commerce, Standard Generalized Markup Language (SGML). Federal Information Processing Standards Publication Number 152, National Institute of Standards and Technology, Washington, DC, September 26, 1988.

US Lynx, Context-Wise. New York, NY, 1991.

van Dam, A., Hypertext 1987 keynote address. Communications of the ACM 31(7):887-895, 1988.

van Dam, A., and Rice, D., On-line text editing: A survey. ACM Computing Surveys 3, pp. 93-114, September 1971.

van Herwijnen, E., Practical SGML. Kluwer Academic Publishers, Boston, MA, 1990.

van Herwijnen, E., and Esteveny, L., CERNDOC - A Document Filing and Retrieval System. Technical report, CERN, 1990.

Weyer, S.A., The Design of a Dynamic Book for Information Search. International Journal of Man-Machine Studies 17(1):87-107, 1982.

Weyer, S.A., and Borning, A., A Prototype Electronic Encyclopedia. ACM Transactions on Office Information Systems 3(1):63-88, 1985.

Wilson, E., Converting an SGML Text to Hypertext (TEI TR3 W6). Computing Laboratory, University of Kent at Canterbury, Canterbury, Kent, England, 1991.

Wirth, N., Lilith: A personal computer for the software engineer. Proceedings of the 5th International conference on Software Engineering (San Diego, CA, March 9-12). ACM, New York, NY, pp. 2-15, 1981.

WordPerfect Corporation, WordPerfect 5.1. Orem, UT, 1993a.

WordPerfect Corporation, Intellitag 1.2. Orem, UT, 1993b.

XSoft, InContext. Palo Alto, CA, 1993.

Yamada, M., Miyasato, T., and Hasuike, K., Mixed mode document processing system. In 1987 IEEE Global Telecommunications Conference (GLOBECOM), New York, NY, pp. 1171-1175, November 1987.

Yankelovich, N., From Electronic Books to Electronic Libraries: Revisiting 'Reading and Writing the Electronic Book.' Hypermedia and Literary Studies, MIT Press, Cambridge, MA, pp. 133-142, 1991.

Zandar Corporation, TagWrite 4 ALCHEMY. Newfane, VT, 1994.

Zellweger, P.T., Active paths through multimedia documents. Proceedings of the International Conference on Electronic Publishing, Document Manipulation, and Typography in Nice, France, Cambridge University Press, Cambridge, NY, pp. 19-34, April 20-22, 1988.

BIOGRAPHICAL SKETCH

Tony V. Harrison was born on March 30, 1957, in Port St. Joe, Florida. He graduated from Port St. Joe High School as an honor graduate in 1975. He was in Who's Who in High Schools in the U.S. during 1975.

After graduation from high school, he enlisted in the United States Marine Corps. During his tour of duty, he visited Spain, Italy, France, Greece, Scandinavia, Sicily, and England. He also spent a fifteen-month extended tour at the marine barracks in Keflavik, Iceland. His primary jobs during his various tours of duty included mortarman, sergeant of the guard, small arms weapons instructor, range noncommissioned officer and a sergeant instructor at the Basic School in Quantico, Virginia. He was honorably discharged as a sergeant (E-5) in August of 1979.

In August, 1979, he entered Gulf Coast Community College. He received an Associate of Arts degree in May, 1981.

In August, 1981, he entered the University of Florida to pursue a bachelor's degree in mechanized agriculture. In August, 1984, he received a Bachelor of Science in Agriculture degree in mechanized agriculture. He worked part-time during his undergraduate career as a student assistant in the Agricultural Engineering Department. He received the American

Society of Agricultural Engineers Blue Ribbon Award for Educational Aids in 1984. During that time, he was assistant editor of the Florida-Caribbean Section Newsletter of ASAE. In August, 1984, he entered the Graduate School as a graduate research assistant in the Agricultural Engineering Department to pursue a Master of Science degree with a specialization in mechanized agriculture. His field of interest during this time was market forecasting and sales management. He was selected to be in the National Dean's List in 1984. He was a member of Alpha Zeta (Agricultural Honor Fraternity); Alpha Mu (Mechanized Agriculture Honor Fraternity); American Society of Agricultural Engineers; American Society of Heating, Refrigerating and Air Conditioning Engineers; and the Mechanized Agriculture Club. He graduated from the University of Florida in 1986 with a Master of Science degree.

Upon completion of his degree, he took a production trainee position with Ralston Purina Company. During the training period, he also was responsible for both the review and then installation/testing procedures for automating the process control operations for all plant machinery via programmable controllers and an IBM Series 1 computer.

Upon completion of the one-year training program in Circleville, Ohio, in 1987 he was assigned as a production shift-supervisor in Minneapolis, Minnesota. Further duties included the automation of both batch and pellet mill production units.

In 1988 he was promoted to production supervisor at the Fort Worth, Texas, plant. His main duties were with total automation of the process control operations, including batch, micro ingredients, pellet mill, liquid, and both loading and unloading equipment.

In 1989, he accepted a position with Southern Agcom, Inc. in Blakely, Georgia. His responsibilities included purchasing, sales, and job bids on contracts for the company. He also was responsible for the automation of office work such as accounting, payroll, and product pricing. He continued in these operations until 1990, when he accepted a position as a graduate assistant at the University of Florida to pursue his doctoral degree in agricultural operations management.

From 1994 to present he is working as a consultant with the Air Force Reserve (AFRES) publications group at Warner Robins, Georgia. His work with AFRES involves replacing printed publications with an infobase on CD-ROM. The author developed a model for AFRES based on the Air Force model. He also developed the rules for converting the Air Force and Air Force Reserve publications into SGML instances based on the model. Currently, he is evaluating the Air Force DTD, providing suggestions to streamline the model for less author involvement. He is also training AFRES staff on how to use all the document conversion tools, from authoring to cutting a CD-ROM.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

J. Wayne Mishoe

J. W. Mishoe, Chairman
Professor of Agricultural
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

D. G. Watson, Cochairman
Associate Professor of
Agricultural Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

R M Peart

R. M. Peart
Graduate Research Professor of
Agricultural Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

H. W. Beck
Assistant Professor of
Agricultural Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

R. L. Kilmer

R. L. Kilmer
Professor of Food and Resource Economics

This dissertation was submitted to the Graduate Faculty of the College of Agriculture and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

May 1995

Jack L. Fry

Dean, College of Agriculture

Dean, Graduate School